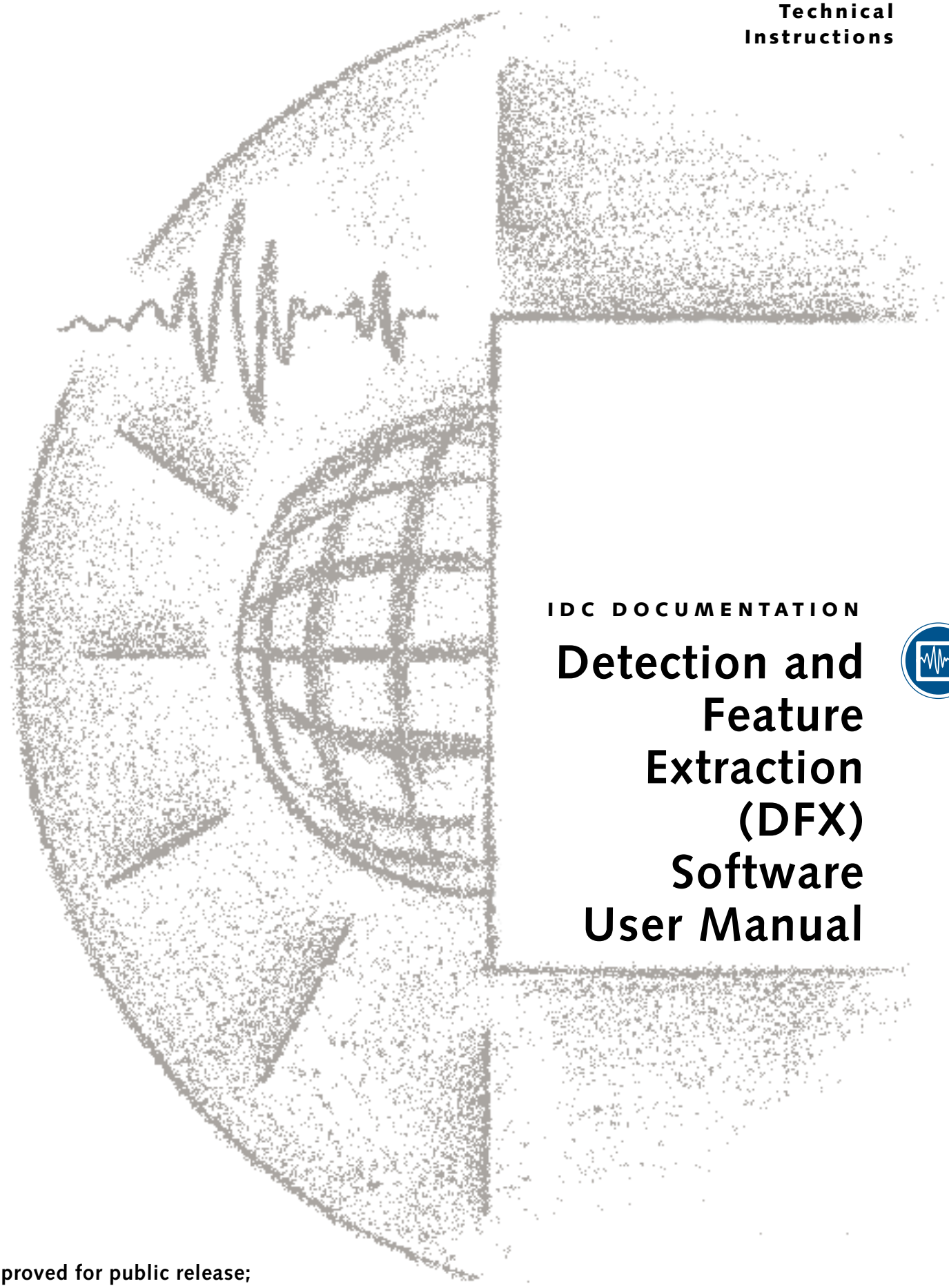
Technical Instructions



IDC DOCUMENTATION

# Detection and Feature Extraction (DFX) Software User Manual

Approved for public release; distribution unlimited

**Notice**

This document was published November 2002 by the Monitoring Systems Operation of Science Applications International Corporation (SAIC) as part of the International Data Centre (IDC) Documentation. Every effort was made to ensure that the information in this document was accurate at the time of publication. However, information is subject to change.

**Contributors**

Ethan Brown, Science Applications International Corporation

**Trademarks**

BEA TUXEDO is a registered trademark of BEA Systems, Inc.
Enterprise is a registered trademark of Sun Microsystems.
Motif 2.1 is a registered trademark of The Open Group.
ORACLE is a registered trademark of Oracle Corporation.
SAIC is a trademark of Science Applications International Corporation.
Scheme In One Defun (SIOD) is a product of Paradigm Associates of Cambridge, Massachusetts.
Solaris is a registered trademark of Sun Microsystems.
SPARC is a registered trademark of Sun Microsystems.
Sun is a registered trademark of Sun Microsystems.
UltraSPARC is a registered trademark of Sun Microsystems.
UNIX is a registered trademark of UNIX System Labs, Inc.

**Ordering Information**

The ordering number for this document is SAIC-02/3044.

This document is cited within other IDC documents as [IDC6.5.3].

# Detection and Feature Extraction (DFX) Software User Manual

## CONTENTS

# Detection and Feature Extraction (DFX) Software User Manual

## FIGURES

# Detection and Feature Extraction (DFX) Software User Manual

## TABLES

# About this Document

This chapter describes the organization and content of the document and includes the following topics:

# About this Document

## PURPOSE

This document describes how to use the Detection and Feature Extraction (*DFX*) software of the International Data Centre (IDC). The software is a computer software component (CSC) of the Automatic Processing, Interactive Processing, and Data Management Computer Software Configuration Items (CSCI) and is identified as follows:

| | |
|---|---|
| Title: | Detection and Feature Extraction |
| Abbreviation: | *DFX* |

## SCOPE

The manual includes instructions for setting up the software, configuring its various applications, and basic troubleshooting. This document does not describe the software's design or requirements. These topics are described in sources cited in "Related Information."

This document provides an understanding of the overall *DFX* configuration in the context of the *DFX* Scheme-based applications. Each of the 14 applications is described to give the user an understanding of the application's role in IDC processing and its key configuration files. The information in this document should allow the user to understand the key application files, and to locate and modify runtime parameters and recipes.

## AUDIENCE

This document is intended for operational personnel responsible for maintaining the *DFX* configuration, adding new stations, and trouble-shooting problems. It should also be helpful to research staff responsible for tuning and other special studies.

## RELATED INFORMATION

The following documents complement this document:

- *Detection and Feature Extraction (DFX)–Scheme Files* [IDC7.1.1]
- *Configuration of IDC Processing Data Files, Revision 1* [IDC6.2.4Rev1]
- *Distributed Application Control System (DACS) Software User Manual, Revision 0.1* [IDC6.5.2Rev0.1]
- *Continuous Data Subsystem CD–1.1* [IDC7.4.1]

See "References" on page 157 for a list of documents that supplement this document. The following UNIX manual (man) pages apply to the existing *DFX* software:

- *DFX* man page

Useful information may also be found in the header portion of the application Scheme files.

## USING THIS DOCUMENT

This document is part of the overall documentation architecture for the IDC. It is part of the Technical Instructions category, which provides guidance for installing, operating, and maintaining the IDC systems. This document is organized as follows:

- Chapter 1: Introduction

  This chapter provides an overview of the software's capabilities, development, and operating environment.

- Chapter 2: Operational Procedures

  This chapter describes how to use the software and includes detailed procedures for startup and shutdown, basic and advanced features, security, and maintenance.

- Chapter 3: Troubleshooting

  This chapter describes how to identify and correct common problems related to the software.

- Chapter 4: Installation Procedures

  This chapter describes first how to prepare for installing the software, then how to install the executable files, configuration data files, database elements, and any Tuxedo files. It also describes how to initiate operation and validate the installation.

- References

  This section lists the sources cited in this document.

- Glossary

  This section defines the terms, abbreviations, and acronyms used in this document.

- Index

  This section lists topics and features provided in this document along with page numbers for reference.

## Conventions

The conventions used in this document are described Table I, which lists typographical conventions.

**TABLE I: TYPOGRAPHICAL CONVENTIONS**

| Element | Font | Example |
|---|---|---|
| database table | **bold** | **sitechan** |
| database table and attribute, when written in the dot notation | | **site.***statype* |
| database attributes | *italics* | *statype* |
| processes, software units, and libraries | | *DFX* |
| user-defined arguments and variables used in parameter (par) files or program command lines | | `station=<`*your station*`>` |
| titles of documents | | *Continuous Data Subsystem* |
| computer code and output | `courier` | `Error: No wfdiscs found` |
| filenames, directories, and websites | | `.../DFX/DFX-evch-hydro.par` |
| text that should be typed exactly as shown | | `DFX par=<`*your par file*`>` |

# Chapter 1: Introduction

This chapter provides a general description of the software and includes the following topics:

# Chapter 1: Introduction

## SOFTWARE OVERVIEW

Figure 1 shows the logical organization of the IDC software. *DFX* applications are part of the Automatic Processing, Interactive Processing, and Data Management CSCIs and are responsible for various tasks in the Station Processing, Post-location Processing, Event Screening, Time-series Analysis, and Data Archiving CSCs. The relationships of *DFX* applications with Automatic Processing, Interactive Processing, and Data Management IDC components are described in detail in Chapter 1 of [IDC7.1.1].

## STATUS OF DEVELOPMENT

*DFX* is a mature component of the IDC system. Its development status is described in detail in [IDC7.1.1].

## FUNCTIONALITY

*DFX* provides a flexible framework that supports a variety of waveform analysis applications. The functionalities of the applications described in this document include automatic seismic, hydroacoustic, and infrasonic signal detection and feature measurement and beam formation for display, event characterization, and data archiving.

Interactive applications such as the Analyst Review Station (*ARS)* rely on *DFX* to provide on-demand feature measurements for analyst-added arrivals, and display-beams for analyst-added events. Table 1 lists the *DFX* applications discussed in this document.

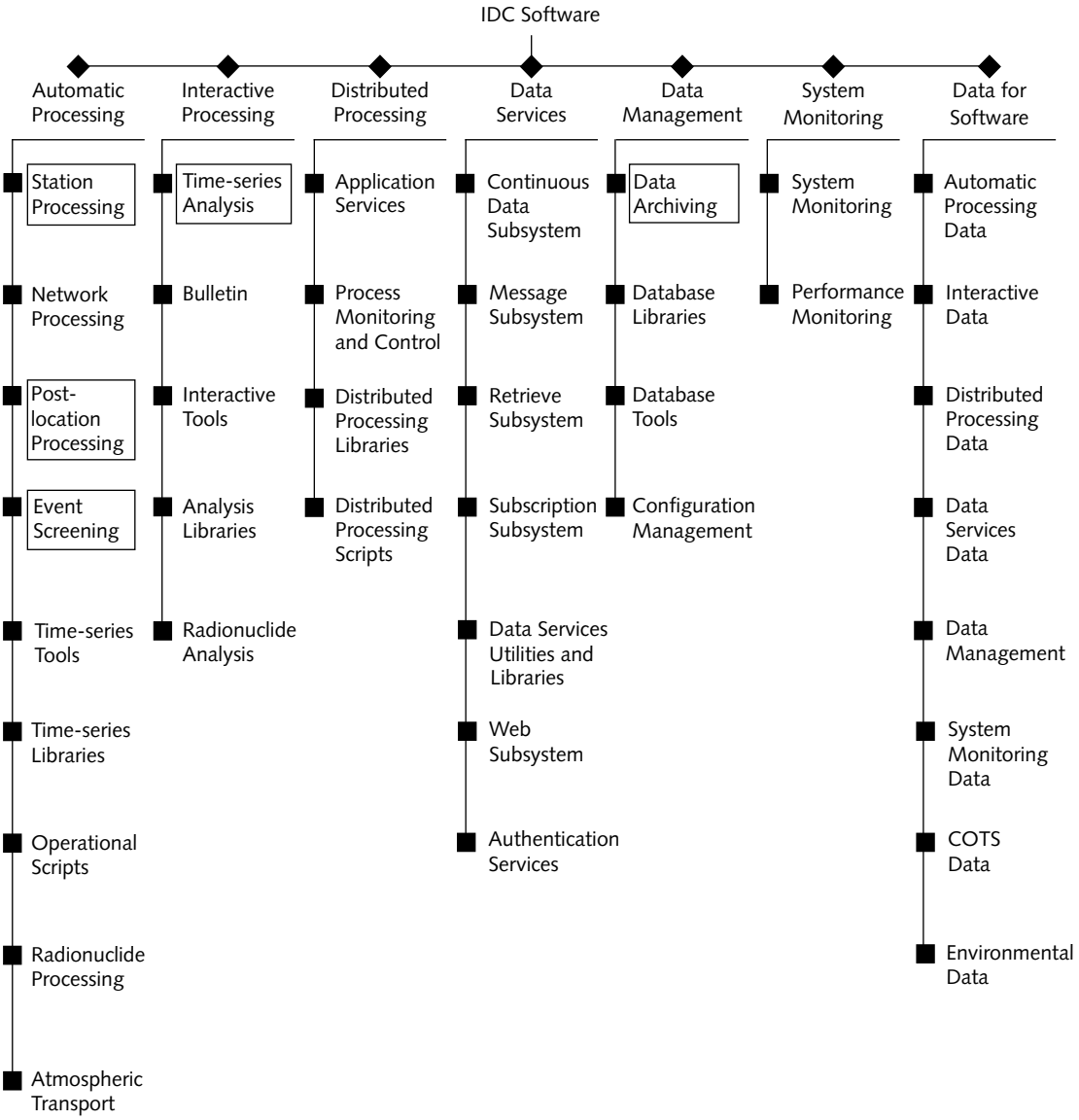



**FIGURE 1. IDC SOFTWARE CONFIGURATION HIERARCHY**

**TABLE 1: DFX APPLICATIONS**

| Application | Mode | Description | Products[1] |
|---|---|---|---|
| *Detection Processing* | Automatic | Seismic signal detection and detection feature measurements. | Arrival features and detection beams |
| *Origin Beam* | Automatic | Origin beams for non-associated stations. | Beams |
| *Beam-on-the-Fly* | Interactive | Interactive origin beam formation used by *ARS*. | Beams |
| *Depth-phase SNR* | Automatic | Pre- and post-detection amplitude ratio | Amplitude measurements |
| *Noise Amplitude* | Automatic | Amplitude measurements for theoretical arrivals. | Amplitude measurements |
| *Seismic Event Characterization* | Automatic | Seismic measurements pertaining to event-type signal characterization. | Amplitude and other discrimination features |
| *Hydro Detection* | Automatic | Hydroacoustic signal detection and detection feature measurements. | Arrival features |
| *Hydro Recall* | Interactive | Detection feature measurements initiated from *ARS*. | Hydroacoustic detection features |
| *Hydro Event Characterization* | Automatic | Hydroacoustic measurements pertaining to event-type signal characterization. | Amplitude measurements |
| *Infra Detection* | Automatic | Infrasonic signal detection and detection feature measurements. | Arrival features and derived beams |
| *Interactive Recall Processing* | Interactive | Seismic detection feature measurements initiated from *ARS*. | Detection feature measurements |
| *QC Stats* | Automatic | Data Quality Control metrics. | QC statistics |
| *Seismic Recall* | Automatic | Post-analysis seismic detection feature measurements. | Detection features |
| *Segment Archiving* | Automatic | Archive origin beams | Beams |

1. For details, see [IDC7.1.1]

## Features and Capabilities

The basis of *DFX* capabilities is the integration of IDC common libraries, database, waveform processing libraries, and *DFX* computational libraries with a Scheme command processing language and generic object (GObj) internal data representation. The various processing tasks are achieved through Scheme applications that access library functionality via Scheme bindings to library functions. Processing recipes, database information, and waveform data are stored as GObjs and provide convenient mapping between the data representations internal and external to *DFX*.

## Performance Characteristics

*DFX* performance characteristics are application specific, because applications may use time-, arrival-, or origin-based time intervals. All applications use site-related database information (**site, sitechan, sensor**, and **instrument** tables), however these have no significant affect on program performance.

### Detection Processing (Seismic)

#### Input Type/Volume

The input waveform volume is dependent on the specified processing interval and the number of station elements. Assuming a typical processing interval of 10 min, a single channel of 20 sample/s data stored as 4-byte integers requires 48 KB. So, for example, a three-component (3-C) station requires 144 KB of data and a 20-element array requires approximately 1 MB of data.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The outputs of detection processing are detection features, which are inserted in database tables, and detection beam waveforms, which are written to files. Detections rates vary between stations but a nominal rate is 10 detections per hour (from 1 to 3 detections per 10 min processing interval). Detections beams are 45 s long so they require 3.6 KB of storage per beam for 20 sample/s data.

#### Processing Time

The processing time depends on the number of station elements and the size of the detection beam set. 3-C stations take approximately 10 s per 10 min processing interval on a Sun Microsystems E450-class computer. Array stations typically take 50 s per 10 min processing interval.

### Origin Beam

#### Input Type/Volume

The *Origin Beam* processing interval is 135 s. A 20-element array station sampled at 20 sample/s stored as 4-byte integers requires 216 KB of data.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The *Origin Beam* produces a single beam with associated **wfdisc** and **wftag** records. The beam is 120 s long, so for 20 sample/s stored as 4-byte integers this gives 9.6 KB of data.

#### Processing Time

*Origin Beam* processing takes less then 15 s per beam on a Sun Microsystems Ultra 60.

### Beam-on-the-Fly

#### Input Type/Volume

The waveform input volume depends on the number of station elements and whether the beam is for teleseismic or regional distance. For a conservative estimate we will assume a 20-element array with a regional beam time duration. *Regional Beam-on-the-Fly* processing creates beams 400 s in length, with an additional 30 s of data preceding and following the beam interval to allow for filter stabilization. Assuming 4 bytes per sample and a sample rate of 20 sample/s, this gives 736 KB of input waveform data.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The type of output data depends on whether the beam is for teleseismic or regional distance. Teleseismic processing produces a single beam with a duration of 180 s. *Regional Beam-on-the-Fly* processing creates three beams, each having a duration of 400 s. Using the same assumptions as the previous section teleseismic processing produces 14.4 KB of beam data, and regional processing produces 96 KB of beam data.

#### Processing Time

Multiple stations may be processed during a single invocation of *Beam-on-the-Fly*. processing time is less than 10 s per station on a Sun Microsystems Ultra 60.

### Depth-phase SNR

#### Input Type/Volume

Input waveforms for depth phases have measurement windows on the order of 5 s with an additional 30 s preceding and following the processing interval for filter stabilization. For a 20-element array station sampled at 20 sample/s, 104 KB of data are required for each arrival.

#### Output Type/Volume

Output is in the form of **amplitude** database records. See [IDC7.1.1] for more detail.

#### Processing Time

*Depth-phase SNR* processing takes approximately 2 s to process one detection on a Sun Microsystems UltraSPARC-IIe.

### Noise Amplitude

#### Input Type/Volume

The input waveform interval is 150 s. A 20-element array station sampled at 20 sample/s stored as 4-byte integers requires 240 KB of data.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The *Noise Amplitude* processing output is in the form of database records, which are described in [IDC7.1.1].

#### Processing Time

A 33-station network with 1 origin takes approximately 5 min to process on a Sun Microsystems Sparc 20 computer.

### Seismic Event Characterization

#### Input Type/Volume

The *Seismic Event Characterization* processes are run using a network of stations and a set of origins that occur in the processing time interval. The data interval length obtained for processing is 175 s for teleseismic processing and 135 s for regional processing. Although a large number of entities are being processed (stations × origins), the volume of input data in memory at any particular time is fairly low because the waveform data are read and processed separately for each station-origin pair. For example, a 20-element array station sampled at 20 sample/s stored as 4-byte integers requires 120 KB of data.

#### Output Type/Volume

*Seismic Event Characterization* processing output is in the form of database records, which are described in [IDC7.1.1].

#### Processing Time

A 40-station network with 10 origins takes less than 11 min to process on a Sun Microsystems Enterprise 4000 computer.

### Hydro Detection

#### Input Type/Volume

The input waveform volume depends on the specified processing interval. Assuming a 10 min processing interval with an 11 min buffer preceding and following the processing interval (32 min total), a single channel of 320 sample/s data stored as 4-byte integers requires 2.5 MB.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The *Hydro Detection* processing output is in the form of database records, which are described in [IDC7.1.1].

#### Processing Time

Processing a 10 min data interval takes approximately 30 s on a Sun Microsystems Enterprise 4000.

### Hydro Recall

#### Input Type/Volume

The waveform data interval for *Hydro Recall* processing is centered on the detection and extends 670 s before and after the detection. Therefore, a channel sampled at 320 sample/s stored as 4-byte integers requires 1.7 MB.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The *Hydro Recall* processing output is in the form of database records, which are described in [IDC7.1.1].

#### Processing Time

Processing time is less than 15 s per detection on a Sun Microsystems Enterprise 4000.

### Hydro Event Characterization

#### Input Type/Volume

The *Event Characterization* processes are run using a network of stations and a set of origins that occur within the processing time interval. The data interval length used for processing is 120 s for processing, plus a buffer to allow for the origin uncertainty and edge effects. Considering these factors, a nominal input data interval length is approximately 250 s. Although a large number of entities are being processed (stations × origins), the volume of input data is fairly low because the waveform data are read and processed separately for each station-origin pair. For example, a station sampled at 320 sample/s stored as 4-byte integers requires 320 KB of data.

#### Output Type/Volume

The hydro event processing output is in the form of database records, which are described in [IDC7.1.1].

#### Processing Time

The time necessary for processing the network of hydroacoustic stations varies depending on the number of origins and unblocked station-to-event water paths. The processing time for 14 unblocked station-origins is 109 s on a Sun Microsystems Ultra 10.

### Infra Detection

#### Input Type/Volume

Input waveform volume depends on the specified processing interval and the number of station elements. With a typical processing interval of 20 min, a 250 s buffer preceding and following the interval, and 4 array elements of 20 sample/s data stored as 4-byte integers, the volume of input data is 136 KB.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

Three beams are output for each station interval, resulting in 0.5 to 3.0 KB of data.

The *Infra Detection* processing output is in the form of database records, which are described in [IDC7.1.1].

#### Processing Time

The processing time depends on the number of potential detections made during the processing interval. Twenty min of data with 1 to 2 detections requires approximately 95 s of processing time on a Sun Microsystems Enterprise 4000.

### Interactive Recall Processing

#### Input Type/Volume

The waveform data interval for *Interactive Recall* processing is centered on, and extends 75 s before and after, the detection. A 20-element array station sampled at of 20 sample/s stored as 4-byte integers requires 240 KB of data.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The *Interactive Recall* processing output is in the form of database records, which are described in detail in [IDC7.1.1].

#### Processing Time

Processing time depends on the type of station and the number of detections being processed. In general, processing takes between 5 and 10 s per detection on a Sun Microsystems Ultra-60.

### QC Stats

#### Input Type/Volume

*QC Stats* is a subprocess of all automatic detection processes and requires no additional input data.

#### Output Type/Volume

*QC Stats* output is in the form of **qcstats** database records.

#### Processing Time

The additional time required for the *QC Stats* portion of detection process is less than a few seconds.

### Seismic Recall

#### Input Type/Volume

The waveform data interval for *Seismic Recall* processing is centered on, and extends 75 s before and after, the detection. A 20-element array station sampled at of 20 sample/s stored as 4-byte integers requires 240 KB of data.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

The *Seismic Recall* processing output is in the form of database records, which are described in [IDC7.1.1].

#### Processing Time

Processing time depends on the type of station and the number of detections being processed. In general, processing takes between 5 and 10 s per detection.

### Segment Archiving

#### Input Type/Volume

*Segment Archiving* currently archives intervals of 330 s. A 20-element array station sampled at a rate of 20 sample/s stored as 4-byte integers requires 528 KB of data. Processing is done over sets of origins and stations. This data volume applies to each origin-station pair.

Other types of input are provided by configuration files and database tables. See [IDC7.1.1] for more detail.

#### Output Type/Volume

*Segment Archiving* outputs a beam 120 s long; therefore, a 20 sample/s channel stored as 4-byte integers contains 9.6 KB of output data.

#### Processing Time

A set of 81 origin-station pairs requires less than 25 min to processes on a Sun Microsystems Enterprise 4000.

## INVENTORY

Each application uses the *DFX* executable, the top-level Scheme files `general.scm`, `intrinsic.scm`, `siod.scm`, `libpar.scm`, `math.scm`, and the `DFX-default.scm`. Scheme files provide many functions common to the applications. In addition, applications require specific Scheme files and possibly travel-time files, which are described in Table 2. Configuration (parameter and recipe) files are described in "Operational Procedures" on page 19, and database tables used by each application are described in [IDC7.1.1].

**TABLE 2: APPLICATION SCHEME FILES**

| Application | Scheme File | Travel-time[1] Data File |
|---|---|---|
| *Detection Processing* | `DFX-detection.scm` | (N/A) |
| *Origin Beam* | `DFX-originbeam.scm` | VMSF |
| *Beam-on-the-Fly* | `DFX-botf.scm` | VMSF[2] |
| *Depth-phase SNR* | `DFX-depth-phase-snr.scm` | (N/A) |
| *Noise Amplitude* | `DFX-noiseamp.scm` | VMSF |
| *Seismic Event Characterization* | `DFX-evch.scm` | VMSF |
| *Hydro Detection* | `DFX-hydro-detection.scm` | (N/A) |
| *Hydro Recall* | `DFX-hydro-recall.scm` | (N/A) |
| *Hydro Event Characterization* | `DFX-evch-hydro.scm` | VMSF |
| *Infra Detection* | `DFX-infra-detection.scm` | (N/A) |
| *Interactive Recall Processing* | `DFX-int-recall.scm` | VMSF |
| *QC Stats* | `DFX-qcstats.scm` | (N/A) |
| *Seismic Recall* | `DFX-recall.scm` | VMSF |
| *Segment Archiving* | `DFX-segarch.scm` | VMSF |

1. Travel-time data for all applicable applications are specified by the top-level (for example, `shared.par`) parameter file `VMSF` parameter.
2. Velocity Model Specification File

## ENVIRONMENT AND STATES OF OPERATION

### Software Environment

*DFX* applications are written in ANSI C and SIOD Scheme. *DFX* has been developed, tested, and run on the Solaris (Version 7) operating system.

Applications obtain some input data from a relational database through the common library *libgdi*. *Libgdi* interacts with several database vendors; however *DFX* has been tested only with an Oracle database.

*DFX* includes some Fortran source files and must be linked with Fortran libraries that are optional to the Solaris distribution (usually located in `/opt/SUNWSpro/lib`).

*DFX* memory usage is application and station-specific, but *Detection Processing* is given as an example. The *DFX* executable with no data loaded into it requires approximately 10 MB.

As described in "Performance Characteristics" on page 5, 20 min of data for a single channel requires approximately 96 KB of memory. Therefore, raw data from a 20-element array requires approximately 2 MB of memory. Each beam is processed separately and has 8 associated vectors such as short-term-average, long-term-average, and snr function. These vectors and other data require approximately 15 MB of memory usage.

### Normal Operational State

All *DFX* applications are invoked from a *tuxshell* process; they have no Inter-Process Communications (IPC) capability. Automatic applications such as *Detection Processing* and various post-analysis recall processing applications are initiated by *tis_server* services (for example, *tis, tis-late*, *tis-recall*). Application-specific information is given in "Normal Operational Invocations" on page 20. The Distributed Application Control System (DACS) Software User Manual [IDC6.5.2Rev0.1] describes these configurations in detail. Interactive applications are invoked through messages from the *ARS* application.

### Contingencies/Alternate States of Operation

*DFX* applications may be run in a stand-alone mode without the use of *tuxshell*. Running an application in this way is most useful for tuning, research activities, and debugging.

Any application can be configured to run stand-alone by examining the *tuxshell* files and manually providing the parameters normally provided by the *tuxshell* par file. Specific examples are shown in "Manual Invocations" on page 27.

# Chapter 2: Operational Procedures

This chapter provides instructions for using the software and includes the following topics:

# Chapter 2: Operational Procedures

## SOFTWARE STARTUP

### Normal Operational Invocations

The context of the *DFX* applications and their place in the overall IDC system are described in this section. All *DFX* applications are invoked through *tuxshell* processes. The steps taken by supporting IDC processes to run each application are described in the following sections.

#### Detection Processing

- The *tis_server* process monitors incoming station data. When more than 80 percent of the waveform data are available for the candidate station interval, the station interval is added to a Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the following par file: …/`app_config/distributed/tuxshell/detpro/tuxshell-DFX.par`
- The *tuxshell* executes the *DFX Detection Processing* application using a combination of static parameters such as the path to the `DFX-detection.par` par file, and interval-specific parameters such as *start-time*, *end-time*, and *station*.
- *DFX* writes the results to the database and beam disk files.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

### Origin Beam

- The *ticron* process provides an interval for origin-based processing. This interval is aded to the *origin beam* Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file …/app_config/distributed/tuxshell/sel3/tuxshell-Beamer.par.
- The *tuxshell* executes the *DFX Origin Beam* application using a combination of static parameters such as the path to the DFX-originbeam.par par file, and interval-specific parameters such as *start-time*, *end-time*, and *network*.
- *DFX* writes its processing results to the database and disk files.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

### Beam-on-the-Fly

- An analyst selects an event and station set from the *ARS* display and clicks the [Beam] toolbar button.
- An IPC message is sent from *ARS* to Tuxedo where it is entered into a Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file …/app_config/distributed/tuxshell/interactive/tuxshell-BOTF.par.
- The *tuxshell* executes the *DFX-botf.scm* application using a combination of static parameters such as the path to the DFX-botf.par par file, and dynamic parameters such as the output **wfdisc** table, database, event time and location, and phase.
- Beams formed by *DFX* processing are written to the temporary **wfdisc** table specified by the *ARS* IPC message.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

- The return status message is sent back to *ARS*, which displays the beams or informs the analyst of a processing error should one occur.

### Depth-phase SNR

*Depth-phase SNR* processing is part of the post-analysis *recall* pipeline.

- An analyst completes an analysis session.
- The **interval** table is updated.
- A *tis_server* process monitoring the **interval** table updates the *Depth-phase SNR* Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file …`/app_config/distributed/tuxshell/recall/tuxshell-DFX-depth-phase-snr.par`.
- The *tuxshell* executes the *Depth-phase SNR* application using a combination of static parameters such as the path to the `DFX-depth-phase-snr.par` par file, and interval-specific parameters such as *start-time*, *end-time*, and *network*.
- *DFX* writes the results into the **amplitude** table.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

### Noise Amplitude

*Noise Amplitude* processing is part of the post-analysis *recall* pipeline.

- An analyst updates the **interval** table following the completion of a data-day.
- A *tis_server* process monitoring the **interval** table adds the interval to the *NoiseAmp* Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file …`/app_config/distributed/tuxshell/recall/tuxshell-DFX-noiseamp.par`.

- The *tuxshell* executes the *DFX Noise Amplitude* application using a combination of static parameters such as the `DFX-noiseamp.par` par `file-path`, and interval-specific parameters such as *start-time*, *end-time*, and *network*.
- *DFX* writes its processing results to the database.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately

### Seismic Event Characterization

*Seismic Event Characterization* is a sub-pipeline of the post-analysis *recall* pipeline.

- An analyst completes an analysis session.
- The **interval** table is updated.
- A *tis_server* process monitoring the **interval** table adds the interval to the *Seismic Event Characterization* Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file `.../app_config/distributed/tuxshell/evch/tuxshell-DFX-evch.par`.
- The *tuxshell* executes the *DFX Seismic Event Characterization* application using a combination of static parameters such as the path to the `DFX-evch.par` par file, and interval-specific parameters such as *start-time*, *end-time*, and *net*.
- *DFX* writes the processing results to the database.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

### Hydro Detection

*Hydro Detection* runs in the same context as *Detection Processing* (Seismic). See "Detection Processing" on page 20.

### Hydro Recall

- An analyst selects the arrival set on the *ARS* display and clicks the recall button.
- Temporary **arrival** and **hydro_features** tables are populated with the selected arrival set information.
- *ARS* sends an IPC message to Tuxedo that in turn creates a *Hydro Recall* queue entry.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par `file` `.../app_config/distributed/tuxshell/interactive/tuxshell-RHYDRO.par`.
- The *tuxshell* executes the *DFX Hydro Recall* application using a combination of static parameters such as the path to the `DFX-hydro-recall.par` par file, and dynamic parameters such as **arrival** and **hydro_features** tables to use for input and output.
- *DFX Hydro Recall* computes feature measurements and stores them in the temporary output tables created by *ARS*.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.
- The exit status is returned to *ARS*, which reads the *DFX* output from the temporary tables.

### Hydro Event Characterization

*Hydro Event Characterization* is part of the post-analysis *recall* pipeline.

- An analyst completes an analysis session.
- The **interval** table is updated.
- A *tis_server* process monitoring the **interval** table adds the interval to the *Hydro Event Characterization* Tuxedo queue.

- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file …/app_config/distributed/tuxshell/evch/tuxshell-DFX-evch-hydro.par.
- The *tuxshell* executes the *DFX Hydro Event Characterization* application using a combination of static parameters such as the path to the DFX-evch-hydro.par par file, and interval-specific parameters such as *start-time*, *end-time*, and *station*.
- *DFX* writes its processing results to the database.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

### Infra Detection

*Infra Detection* runs in the same context as *Detection Processing* (Seismic). See "Detection Processing" on page 20.

### Interactive Recall Processing

- An analyst selects the arrival set on the *ARS* display and clicks the recall button.
- Temporary **arrival** table is populated with the selected arrival set. Temporary output tables are created if necessary.
- *ARS* sends an IPC message to Tuxedo that in turn creates a Tuxedo queue entry.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file …/app_config/distributed/tuxshell/interactive/tuxshell-RSEISMO.par.
- The *tuxshell* executes the *DFX Interactive Recall* application using a combination of static parameters such as the path to the *DFX Detection* par file, and interval-specific parameters such as *start-time*, *end-time*, and *station*.

- *DFX Interactive Recall* processing computes arrival feature measurements and stores them in the temporary output tables created by *ARS*.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.
- The exit status is returned to *ARS*, which reads the *DFX* output from the temporary tables.

### QC Stats

*QC Stats* runs in the context of the Seismic, Hydroacoustic, and Infrasonic (S/H/I) *Detection Processing* applications as described in "Detection Processing" on page 20.

### Seismic Recall

*Seismic Recall* is part of the post-analysis *recall* pipeline (not to be confused with *Interactive Seismic Recall*).

- An analyst completes an analysis session.
- The **interval** table is updated.
- A *tis_server* process monitoring the **interval** table updates the *Seismic Recall* Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file `…/app_config/distributed/tuxshell/recall/tuxshell-DFX-recall.par`.
- The *tuxshell* executes the *DFX Seismic Recall* application using a combination of static parameters such as the path to the `DFX-recall.par` par file, and interval-specific parameters such as *start-time*, *end-time*, and *network*.
- *DFX* writes its output to the database.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

### Segment Archiving

*Segment Archiving* is part of the post-analysis *segment archive* pipeline.

- The **interval** table is updated.
- A *tis_server* process monitoring the **interval** table updates the *Segment Archiving* Tuxedo queue.
- The Tuxedo queue entry causes a *tuxshell* invocation using the configuration given in the par file …`/app_config/distributed/tuxshell/segarch/tuxshell-DFX-segarch.par`.
- The *tuxshell* executes the *DFX Segment Archiving* application using a combination of static parameters such as the path to the `DFX-recall.par` par file, and interval-specific parameters such as *start-time*, *end-time*, and *networks*.
- *DFX* writes its output to the database.
- The termination status of *DFX* is captured by *tuxshell*, which then updates the Tuxedo queue appropriately.

## Manual Invocations

Manual invocations of *DFX* applications avoid the processing chains described in the previous section, which is useful for tuning, research, bug identification, and bug-fix verification.

If manual invocation is used to identify an `errobj`-type failure (an error in the Scheme processing), then set the parameter `processing-mode=interactive` to keep the application in the Scheme interpreter when the failure occurs so you can obtain more information about the nature of the problem. The default setting of *processing-mode* is `"operational,"` which causes *DFX* to exit when processing completes either normally or because of an error.

You must run these applications from a correctly configured environment. For example, the environment variables such as *IMSPAR* and *SCHEMEPATH* must be set properly.

The following sections provide sample par files to run *DFX* applications manually.

### Detection Processing

The *Detection Processing tuxshell* provides the *start-time*, *end-time*, and *station* for the *DFX Detection Processing* application. The following par file configuration is recommended to run *Detection Processing* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

```
start-time=<epoch-start-time>
end-time=<epoch-end-time>
station=<your station>
par=<path to application par file>
(for example, /cmss/config/app_config/DFX/DFX-detection.par)

database-account=<your database>
processing-mode=interactive
verbosity=5
```

The application is run using the following command:

```
DFX par=<your par file>
```

### Origin Beam

The *Origin Beam tuxshell* provides the *start-time*, *end-time*, and *network* for the *DFX Origin Beam* application. The following par file configuration is recommended to run *Origin Beam* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

start-time=<*epoch-start-time*>
end-time=<*epoch-end-time*>
net=<*the affiliation network of stations to process*>
par=<*path to application par file*>
(for example, /cmss/config/app_config/DFX/DFX-originbeam.par)

database-account=<*your database*>
processing-mode=interactive
verbosity=5

The application is run using the following command:

DFX par=<*your par file*>

### Beam-on-the-Fly

The *Beam-on-the-Fly tuxshell* provides the station and event information for the DFX-botf.scm application. The following par file configuration is recommended to run *Beam-on-the-Fly* manually. You will need to customize these parameters to form the target beam. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

dbperm=<*your database*>
lat=<*event-lat*>
lon=<*event-lon*>
depth=<*event-depth*>
time=<*event-origin-time*>
dir=<*base-directory-for-output-beams*>
wfdisc_tbl=<*wfdisc-table-for-output-beams*>
phase=<*beam-to-this-phase-slowness, for example, P*>
sta=<*Station*>
qc_sta=<*set to sta if you want data QC, NONE otherwise*>
excluded_chans=<*STA/CHAN list to exclude, NONE includes all channels*>
processing-mode=interactive

verbosity=5
par=<*path to application par file*>
(for example, /cmss/config/app_config/DFX/DFX-botf.par)

The application is run using the following command:

DFX par=<*your par file*>.

### Depth-phase SNR

The *Depth-phase SNR tuxshell* provides the *start-time*, *end-time*, and *network* for the *DFX Depth-phase SNR* application. The following par file configuration is recommended to run *Depth-phase SNR* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

start-time=<*epoch-start-time*>
end-time=<*epoch-end-time*>
net=<*the affiliation network of stations to process*>
par=<*path to application par file*>
(for example, /cmss/config/app_config/DFX/DFX-depth-phase-snr.par)

database-account=<*your database*>
processing-mode=interactive
verbosity=5

The application is run using the following command:

DFX par=<*your par file*>

### Noise Amplitude

The *NoiseAmp tuxshell* provides the *start-time*, *end-time*, and *network* for the *DFX Noise Amplitude* application. The following par file configuration is recommended to run *Noise Amplitude* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

```
start-time=<epoch-start-time>
end-time=<epoch-end-time>
net=<the affiliation network of stations to process>
par=<path to application par file>
```
(for example, `/cmss/config/app_config/DFX/DFX-noiseamp.par`)

```
database-account=<your database>
processing-mode=interactive
verbosity=5
```

The application is run using the following command:

```
DFX par=<your par file>
```

### Seismic Event Characterization

The *Seismic Event Characterization tuxshell* provides the *start-time*, *end-time*, and *network* for the *DFX Seismic Event Characterization* application. The following par file configuration is recommended to run *Seismic Event Characterization* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

`start-time=`*<epoch-start-time>*
`end-time=`*<epoch-end-time>*
`net=`*<the affiliation network of stations to process>*
`par=`*<path to application par file>*
(for example, `/cmss/config/app_config/DFX/DFX-evch.par`)

`database-account=`*<your database>*
`processing-mode=interactive`
`verbosity=5`

The application is run using the following command:

`DFX par=`*<your par file>*

### Hydro Detection

*Hydro Detection* processing is run in the same manner as *Detection Processing* (Seismic). The station specification file settings used by `DFX-site-detection.par` enable the hydroacoustic-specific processing.

### Hydro Recall

The *Hydro Recall tuxshell* provides the **arrival**, **hydro_features**, and **detection** table names, and the database-account for the *DFX Hydro Recall* application. The following par file configuration is recommended to run *Hydro Recall* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Note that *Hydro Recall* processes all detections in the specified **arrival** table, so it is best to create your own test table and populate it with one or several test **arrival** records. The **hydro_features** table must contain records for measurements corresponding to the **arrival** records. If you are creating your own test data, the *hydro_features* required for input are *onset_time*, *termination_time*, *low_cut*, *high_cut*, *ford*, *ftype*, and *fzp*. Use *ARS* to populate temporary input tables containing the arrivals you want to process. Copy the contents of the tables to your own test tables.

Sample par file:

```
arrival-table=<arrival table name>
hydro_features-table=<hydro_features table name>
detection-table=<detection table name>
par=<path to application par file>
(for example, /cmss/config/app_config/DFX/DFX-hydro-
recall.par)

database-account=<your database>
processing-mode=interactive
verbosity=5
```

The application is run using the following command:

```
DFX par=<your par file>
```

### Hydro Event Characterization

The *Hydro Event Characterization tuxshell* provides the *start-time*, *end-time*, and *network* for the *DFX Hydro Event Characterization* application. The following par file configuration is recommended to run *Hydro Event Characterization* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

```
start-time=<epoch-start-time>
end-time=<epoch-end-time>
net=<the affiliation network of stations to process>
par=<path to application par file>
(for example, /cmss/config/app_config/DFX/DFX-evch-hydro.par)

database-account=<your database>
processing-mode=interactive
verbosity=5
```

The application is run using the following command:

```
DFX par=<your par file>
```

### Infra Detection

The *Infra Detection tuxshell* provides the *start-time*, *end-time*, and *station* for the *DFX Infra Detection* application. The following par file configuration is recommended to run *Infra Detection Processing* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

```
start-time=<epoch-start-time>
end-time=<epoch-end-time>
station=<your station>
par=<path to application par file>
(for example, /cmss/config/app_config/DFX/DFX-infra-
detection.par)

database-account=<your database>
processing-mode=interactive
verbosity=5
```

The application is run using the following command:

```
DFX par=<your par file>
```

### Interactive Recall Processing

The *Interactive Recall tuxshell* provides the input arrival and output table names, and the database-account for the *DFX Interactive Recall* application. The following par file configuration is recommended to run *Interactive Recall* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

*Recall Processing* will be done on all detections in the specified **arrival** table; therefore, it is best to create your own test table and populate it with one or several test **arrival** records. Use *ARS* to populate temporary input tables containing the arrivals you want to process. Copy the contents of the tables to your own test tables.

Sample par file:

```
olat=<latitude of associated origin>
olon=<longitude of associated origin>
odepth=<depth of associated origin>
assoc-table=<assoc table name; used only for depth-phase-snr measure-
ments>
par=<path to application par file>
(for example, /cmss/config/app_config/DFX/DFX-int-recall.par)

database-account=<your database>
processing-mode=interactive
verbosity=5
```

The application is run using the following command:

```
DFX par=<your par file>
```

### QC Stats

*QC Stats* is run from within the S/H/I detection applications.

### Seismic Recall

The *Seismic Recall tuxshell* provides the *start-time*, *end-time*, and *network* for the *DFX Seismic Recall* application. The following par file configuration is recommended to run *Seismic Recall* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

```
start-time=<epoch-start-time>
end-time=<epoch-end-time>
net=<the affiliation network of stations to process>
par=<path to application par file>
(for example, /cmss/config/app_config/DFX/DFX-recall.par)
```

database-account=<*your database*>
processing-mode=interactive
verbosity=5

The application is run using the following command:

DFX par=<*your par file*>.

### Segment Archiving

The *Segment Archiving tuxshell* provides the *start-time*, *end-time*, and *network* for the *DFX Segment Archiving* application. The following par file configuration is recommended to run *Segment Archiving* manually. You may need to customize other parameters such as database table names or parameters related to your testing or tuning purposes.

Sample par file:

start-time=<*epoch-start-time*>
end-time=<*epoch-end-time*>
net=<*the affiliation network of stations to process*>
par=<*path to application par file*>
(for example, /cmss/config/app_config/DFX/DFX-segarch.par)

database-account=<*your database*>
processing-mode=interactive
verbosity=5

The application is run using the following command:

DFX par=<*your par file*>

## SOFTWARE SHUTDOWN

All of the *DFX* applications are ephemeral, with automatic application processing durations typically not exceeding 10 min and interactive applications typically not exceeding 30 s. However, should it be necessary to terminate *DFX*, the methods for shutting down the automatic and interactive applications are provided in the following sections.

### Automatic Application Shutdown Procedures

#### Tuxedo Shutdown

*DFX* applications are run through the DACS Tuxedo system (see "Normal Operational Invocations" on page 20 for more detail). Shutting down a Tuxedo-invoked application is accomplished through the *Tuxpad* shutdown facilities. *Tuxpad* shutdown procedures are described in [IDC6.5.2Rev0.1]. The procedure for shutting down a specific *DFX* application from the command line (that is, without *Tuxpad*) is:

1. Execute `tmadmin`
2. Enter > `shutdown -i` *SRVID*

where *SRVID* is the server number to be shut down and is the logical reference to a *tuxshell* server for a specific *DFX* application.

#### Shell Shutdown

To shut down a *DFX* application outside of Tuxedo, use the following procedures. (We recommend that you shut down applications within Tuxedo because a terminated process may be re-started by the DACS system.)

1. Find the machine on which the application is running.
2. Find the process ID.
3. Execute the following command (which sends `SIGTERM` to the process):
   `% kill` <*process ID*>

4. Check to see if the process terminated. If the process did not die, then execute the following command (which sends `SIGKILL` to the process):
   `% kill -9` *<process ID>*

### Interactive Application Shutdown

When an interactive application is initiated from *ARS* an "Abort" dialog box is displayed. Pressing this button will cause *ARS* to stop waiting for a return from the *DFX* application, and the *DFX* application should terminate when complete (typically less than 20 s). If the *DFX* application does not terminate in a reasonable amount of time and you suspect the process is blocked, follow the steps in the "Shell Shutdown" section.

## BASIC PROCEDURES

*DFX* is not typically run by a user. See "Manual Invocations" on page 27 for instructions on running *DFX* as stand-alone applications.

## MAINTENANCE

To successfully maintain the *DFX* applications it is necessary to understand the relationships between the Scheme application file and the various configuration files. In addition, knowing the location of the log files generated during operations is important for monitoring the general application state-of-health and to help identify problems.

The following sections assist in maintenance tasks. The first section describes recipe files and the second section describes their relationships in the context of each *DFX* application.

### Recipe Types and Formats

Each type of *DFX* processing uses a set of parameters that controls aspects of the processing. These parameters are typically grouped into recipe files and are organized in directories that contain recipes specific to a type of processing. For exam-

ple, a directory containing amplitude measurement recipes might contain a file specific to arrival-based amplitude measurements and another file specific to noise-based amplitude measurements. The sections that follow describe the parameters that make up each recipe type and the directories where they are located. The heading names of the sections match the configuration subdirectory names. For example, the …`config/app_config/DFX/hydro` subdirectory contains hydroacoustic recipe parameters and the section describing these parameters is called "Hydro Recipe".

Some recipe parameters are not specified in the configuration files; default values for these parameters, set in the *DFX* or Scheme source, are used during processing. Time units are seconds (s) unless otherwise noted. Slowness units are s/km unless otherwise noted.

### Amp Recipe

The amp recipe parameters control the amplitude measurement computations and provide a list of measurement types as a parameter table. The parameter descriptions are separated into general processing parameters and amplitude recipe parameters (see Tables 3 and 4).

The amp recipe parameters are contained in files in the …`config/app_config/DFX/amp` directory.

**TABLE 3: GENERAL AMP PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *amp-min-sta-fraction* | float | `0.1` | Minimum fraction of available non-masked beam elements to allow an amplitude measurement in the interval specified by the parameter *amp-min-sta-len-fraction*.<br>These parameters provide quality control thresholds. |
| *amp-min-sta-len-fraction* | float | `0.9` | Interval used to check the *amp-min-sta-fraction* threshold (s). |

**TABLE 3: GENERAL AMP PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *amp-decimation-fraction* | float | `0.0` | This parameter is used to screen out the smaller wiggles in the signal during the peak-trough identification. Any peak-trough pair having an amplitude less than the product of *amp-decimation-fraction* and the maximum peak-trough amplitude is screened out. |
| *amp-inst-resp-units* | string | `"d"` | Instrument response correction units. Valid values are `"d"` (displacement), `"v"` (velocity), and `"a"` (acceleration). See the *libresponse* manual section for more information. |
| *amp-inst-resp-type* | string | `"theoretical"` | Instrument correction response type: `"theoretical"` or `"measured"`. See the *libresponse* manual section for more information. |
| *amp-filt-rolloff* | float | `20.0` | Filter rolloff threshold (dB). No filter correction is made for frequencies where the filter rolloff exceeds this threshold. This parameter is used to screen measurements with periods outside of the filter passband. |
| *amp-use-interp-period* | int | `0` | Period interpolater flag. If 1, the period value is adjusted by fitting a sine wave to the signal samples. The objective is to improve period estimation for signal frequencies that approach the Nyquist frequency. |

**TABLE 4: AMPREC RECIPE TABLE PARAMETERS**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *name* | string | "-" | Amplitude recipe name. The recipe name helps associate amplitude recipes with processes. An **amprec** table can contain any number of recipes, but a process may only need a subset of them. Setting a top-level parameter to specify the relevant amplitudes (for example, `amp-measure-set= A5/2,N`) allows the application Scheme code to extract and process the necessary recipes. The *name* may be non-unique in the table, but the *name-phase* pairs must be unique. |
| *phase* | string | "-" | Phase on which to measure the amplitude. |
| *tirec* | string | "-" | Name of the time-interval recipe to use for this measurement (see "TI Recipe" on page 77). |
| *beamrec* | string | "-" | Name of the beam recipe to use for this measurement (see "Beam Recipe" on page 43). |
| *mtype* | string | "-" | Measurement type. See the *DFX* man pages for a list of mtypes. |
| *mvalue* | float | `-1.0` | *mtype*-specific value. See the *DFX* man pages for a description of mvalues and their units for each *mtype*. |
| *irsep* | int | `1` | Instrument response removal flag; `1`: remove response, `0`: do not remove response. |
| *fresp* | int | `1` | Filter response removal flag; `1`: remove response, `0`: do not remove response. |

### Amp3c Recipe

The amp3c recipe parameters control the 3-C amplitude measurement computations and provide the list of frequency bands for measurement as a parameter table. The parameter descriptions are separated into general processing parameters and filter band recipe table-parameters (see Tables 5 and 6).

The amp3c recipe parameters are located in the …`config/app_config/DFX/amp3c` directory and in the station par files in …`config/station_specs/`.

**TABLE 5: GENERAL AMP3C PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *amp3c-sta* | string | NULL | Station on which to measure the amplitude. This parameter is currently set in the station specification par file: …`config/station_specs/`<*STA*>`.par`. |
| *amp3c-chan-list* | string | NULL | Channels specifying the three components for measurement, for example, `"sz,sn,se"`. This parameter is currently set in the station specification par file: …`config/station_specs/`<*STA*>`.par`. |

**TABLE 6: AMP3C FILTER BAND TABLE PARAMETERS**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *band* | string | none | A unique key for identifying a particular filter band specification. |
| *nlead* | float | `5.1` | Lead time before the measurement reference time (typically the arrival time) to start the noise measurement interval (s). |
| *nlen* | float | `5.0` | Length of the noise measurement interval (s). |

**TABLE 6: AMP3C FILTER BAND TABLE PARAMETERS (CONTINUED)**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *slead* | float | `0.1` | Lead time before the measurement reference time (typically the arrival time) to start the signal measurement interval (s). |
| *slen* | float | `5.0` | Length of the signal measurement interval (s). |
| *flo* | float | `1.0` | Filter low cut-off (Hz). |
| *fhi* | float | `3.0` | Filter high cut-off (Hz). |
| *ftype* | string | `"BP"` | Filter type. See *ftype* in "Table 8: Beam-recipe Table Parameters" on page 46 for a list of types. |
| *ford* | int | `3` | Filter order. |
| *zp* | int | `0` | Filter zero phase flag; `0`: non zero-phase (causal), `1`: zero-phase (acausal). |
| *taper* | float | `0.1` | Fraction of the filter onset to taper. |
| *fonset* | float | `10.0` | Filter onset buffer length (s). See *beam-filter-onset-len* in "Table 7: Common Beam Processing Parameters" on page 45. |

### Beam Recipe

The beam recipe files are located in the subdirectories of …`config/app_config/DFX/beam/`. However, many files in the …`config/app_config/DFX/beam` directory are not beam recipes, rather they are group definition files that describe which station elements belong to which station element groups. The beam recipes used in the various types of processing are located in the following sections. The beam recipe parameter descriptions follow.

### Detection Beam Files

The beam recipes used during detection-type processing are located in the directory …config/app_config/DFX/beam/detection.

### Evch Beam Files

The beam recipes used during event characterization processing are located in the directory …config/app_config/DFX/beam/evch.

### Hydro Beam Files

The beam recipes used during hydroacoustic processing are located in the directory …config/app_config/DFX/beam/hydro.

### Origin Beam Files

The beam recipes used during origin-based processing are located in the directory …config/app_config/DFX/beam/originbeam.

### Beam Time Delay Files

The …config/app_config/DFX/beam/tdcorr directory contains beam time delay files (currently only for station NOA). The time delay files contain information necessary for NORSAR large-aperture array processing techniques.

Tables 7 and 8 describe the common beam parameters and the beam-recipe-specific parameters. Not all beam recipe table columns need be included in a beam recipe set. For example, the large-aperture correction parameters are included only in some NORSAR subarray recipes.

**TABLE 7: COMMON BEAM PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *beam-sta* | string | `"-"` | Station array name for the beam recipe set, for example, ARCES, CMAR. This name is commonly set to the value of the `sta` parameter (`beam-sta=$(sta)`), which is set in the station's beam element group par file. |
| *beam-taper-fraction* | float | `0.5` | Fraction of the *beam-filter-onset-length* to taper during beamform filtering. |
| *beam-filter-onset-len* | float | `10.0` | Interval length to add before the beam (both before and after for zero-phase) for filter stabilization (s). Increase the default value for long-period processing, as the default value is most valid for short-period processing. |
| *beam-adjust-group-delay* | int | `0` | Flags a time shift of the beam to account for the filter group delay; `0`: do not apply delay, `1`: apply delay. |
| *beam-demean* | int | `1` | Flags a demean of each data channel before beamforming; `0`: do not demean, `1`: demean. |
| *beam-time-delay-file* | string | NULL | Location of the time delay file for NORSAR large-aperture array processing (used only for NOA stations). This parameter is set in the station's beam element group par file (see "Beam Element Groups" on page 48). |

**TABLE 8: BEAM-RECIPE TABLE PARAMETERS**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *name* | string | `"-"` | Beam recipe identification string (unique within the beam recipe set). Whitespace characters may not be used. |
| *type* | string | `"-"` | Type of beam. The recognized types of beam are `"inc"`, `"coh"`, and `"rms"` for incoherent, coherent, and root mean squared, respectively. See the *libbeam* documentation for more information on beam types. |
| *rot* | string | `"-"` | Type of beam rotation. The recognized values of beam rotation are `"no"`, `"rad"`, and `"tang"`. Beam rotation is applied only to horizontal element beams to create a beam steered radially or tangentially at the station with respect to an event location. |
| *std* | int | `0` | Flag to indicate that this is a "standard" beam that will be used for an sbsnr (standard beam snr) measurement. These measurements are stored in the **amplitude** database table. This parameter is used only during *Detection Processing*. |
| *snr* | float | `-1.0` | Detection threshold value. A detection is declared when the beam's detection statistic value (for example, snr) exceeds this threshold and is in accordance with other detection parameters. This value is used during processing to flag the beam as a detection beam when the value is greater than 0.0. |

**TABLE 8: BEAM-RECIPE TABLE PARAMETERS (CONTINUED)**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *azi* | float | `-1.0` | Azimuth used to steer the beam, in degrees east of north, typically used for detection beams. During *Origin Beam* processing a valid (> 0) value supersedes the computed station-to-event azimuth. |
| *slow* | float | `-1.0` | Slowness used to steer the beam (s/km). During *Origin Beam* processing a valid (non -1.0) value supersedes the computed station-to-event slowness. |
| *phase* | string | `"-"` | Phase to use for computing the beam slowness, typically used during origin beam computation. A valid "slow" parameter will supersede the phase-computed value during processing. |
| *flo* | float | `-1.0` | Low filter cut-off frequency (Hz). |
| *fhi* | float | `-1.0` | High filter cut-off frequency (Hz). |
| *ford* | int | `-1` | Filter order. |
| *ftype* | string | `"-"` | Filter type. One of `"BP"`, `"BR"`, `"HP"`, `"LP"`, or `"NO"` for bandpass, bandreject, highpass, lowpass, or no filter. |
| *zp* | int | `0` | Zero phase filter flag; `0`: non zero-phase (causal), `1`: zero-phase (acausal). |
| *delay* | string | `"normal"` | Type of large-aperture array correction to use. Allowable values are `"normal"`, `"correct"`, and `"absolute"` (see *beam-time-delay-file* in "Table 7: Common Beam Processing Parameters" on page 45). |

**TABLE 8: BEAM-RECIPE TABLE PARAMETERS (CONTINUED)**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *refsta* | string | NULL | Reference station to use during large-aperture array correction. |
| *group* | string | N/A | Beam element group name. This parameter is used to specify the array elements that will contribute to the beam. See Beam Element Groups. |

### Beam Element Groups

The beam element groups file contains specific groups of array elements that are referenced in the beam recipes (see Tables 9 and 10). The element group members are the array elements that will be used to form the beam. Vertical components and horizontal components are typically organized into their own groups, as are long-period elements. The beam element group par files are located in the `…config/app_config/DFX/beam` directory. They may contain general station-wide beam recipe parameters such as *beam-sta* (see "Table 7: Common Beam Processing Parameters" on page 45), as the station beam element groups are referenced from the process-specific beam recipe files.

The beam element group par files are comprised of a single column table listing the element group names, followed by the element group tables.

**TABLE 9: BEAM-GROUP TABLE PARAMETERS**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *group* | string | none | Name of a beam element group parameter table (must be non-whitespace characters). |

**TABLE 10: BEAM ELEMENT TABLE PARAMETERS**

| Table Par | Type | Default | Description |
|---|---|---|---|
| *sta* | string | NULL | Station element name (for example, `CM01`). |
| *chan* | string | NULL | Station element component (for example, `sz`, SHZ). |
| *wgt* | float | `1.0` | Element weight to be applied during beamforming. |
| *atd* | float | `0.0` | Element time delay to use when the large aperture array delay correction type is set to `"absolute"` (s) (see "Table 8: Beam-recipe Table Parameters" on page 46). This parameter is usually omitted from the element tables. |

### Complexity Recipe

The complexity recipe parameters control the complexity measurement computation. Complexity is a ratio of the energy in the coda of a signal to the energy in the signal itself (see Table 11). The complexity par files are located in the directory `...config/app_config/automatic/DFX/complexity/`.

**TABLE 11: COMPLEXITY PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *complexity-signal-len1* | float | `5.0` | Length of signal window (s). The window starts at the detection time. |
| *complexity-signal-len2* | float | `30.0` | Length of coda window (s). The window starts at the end of the *complexity-signal-len1* window. |
| *complexity-noise-len* | float | `10.0` | Length of the pre-detection noise window (s). |

### Detection Recipe

The detection recipe parameters control the signal detection computation (Table 12). The detection par files are located in the directory …`config/app_config/automatic/DFX/det`.

**TABLE 12: DETECTION PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *det-min-sta-fraction* | float | `0.1` | Fraction of available beam elements to allow a detection on this beam. For example, if a beam is constructed from 10 elements and `det-min-sta-fraction=0.4` then there must be data available from at least 4 elements. Element data are unavailable if there are no element data (**wfdisc**) for the particular processing interval, or if the data are masked due to data quality control (QC). |
| *det-stav-measure* | string | `"absolute"` | Method for computing the short-term average value. Values are `"absolute"` or `"square"` for a sample's absolute value or squared value. |
| *det-stav-method* | string | `"sliding"` | Type of short-term average computation; either `"sliding"` or `"recursive"`. The `"sliding"` method is the simple average over the window by including the scaled leading-edge sample value in the new window and removing the scaled trailing-edge sample value from the last window. The `"recursive"` method includes the scaled leading-edge sample from the new window and removes the short-term average value from the last window. |
| *det-stav-len* | float | `1.0` | Length of the short-term average window (s). |

**TABLE 12: DETECTION PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *det-stav-fraction* | float | `1.0` | Percentage of the short-term-average window that must have at least *det-min-sta-fraction* elements available to allow a detection. For example, if `det-stav-len=2`, `det-stav-fraction=0.5`, `det-min-sta-fraction=0.4`, and there are 10 elements in the beam, then if there are at least 4 elements available for 1 s of the 2 s window, a detection can be made on the beam. |
| *det-ltav-len* | float | `30.0` | Length of the long-term average window (s). |
| *det-ltav-fraction* | float | `0.9` | Percentage of the long-term-average window that must have at least *det-min-sta-fraction* elements available to allow a detection. See *det-stav-fraction* for an example. |
| *det-snr-method* | string | `"standard"` | Method used to compute the snr, values `"standard"`, `"z"`, or `"logz"`. `"standard"` is the sta/lta measure. `"z"` is the value (stav – ltav)/sqrt(var(stav)), and `"logz"` is (log(stav) – log(ltav))/sqrt(var(log(stav))) |
| *det-trigger-ratio* | string | `"2,4,4"` | Relationship of a number of samples that must exceed the detection threshold to declare a detection and to declare the detection end. For the default values, a detection is declared when 2 out of 4 (the first 4) samples exceed the threshold. The detection ends when 2 out of 4 (the second 4) samples drop below the value of [*threshold* × *det-sustain-trigger-state-fraction*]. |

**TABLE 12: DETECTION PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *det-sustain-trigger-state-fraction* | float | `1.0` | Adjustment to the detection threshold when the beam is in a triggered state. The beam becomes untriggered when it drops below the value of [*threshold* × *det-sustain-trigger-state-fraction*] in the manner described for the *det-trigger-ratio* parameter. |
| *det-max-stav-window-len* | float | `4.0` | Time interval following a detection to measure the maximum short-term average feature value (s). If this value exceeds *det-beam-assoc-len*, then its value is used for the interval. |
| *det-beam-assoc-len* | float | `3.0` | Detection triggers within this time range are grouped as the same detection (s). |
| *det-min-detection-interval* | float | `4.0` | Minimum interval following a detection before another detection may be declared (s). If the value of *det-beam-assoc-len* exceeds this value, then its value is used. |
| *det-min-triggered-beams* | int | `1` | At least this many beams must be triggered to declare a detection. |
| *det-min-adjacent-beams* | int | `1` | Minimum number of triggered beams adjacent in slowness to the best (highest snr) beam to declare a detection. See *det-max-adjacent-beam-rad*. |
| *det-max-adjacent-beam-rad* | float | `0.2` | Maximum slowness radius that defines beams as adjacent (s/km). |
| *det-deltime* | float | `-1.0` | Detection *deltime* override value (s). When set to a value > 0 this value is used as the detection's *deltime* value in place of the computed value. |

**TABLE 12: DETECTION PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *det-amprec-for-arrival* | string | NULL | Amplitude recipe name for the detection amplitude measure (see "Table 4: Amprec Recipe Table Parameters" on page 41). |
| *det-amprec-list* | GObj | | List of amplitude recipe names specifying the amplitude set to measure for a detection (see "Table 4: Amprec Recipe Table Parameters" on page 41). |
| *det-screen-max-slow* | float | `999.0` | Screen out any detections with a slowness greater than this value (s/km) |
| *det-screen-max-fkqual* | float | `999.0` | Screen out any detections with an f-k quality greater than this value. The `fkqual` is a value between 1 and 4 that describes the difference in dB between the two highest peaks; `1`: more than 6 dB difference, `2`: 4–6 dB difference, `3`: 2–4 dB difference, and `4`: 0–2 dB difference. |
| *det-screen-min-fstat* | float | `-1.0` | Screen out detections with *fstats* below this value. The *fstat* is the measure of amount of power in the f-k peak compared with the total power in the data. |
| *det-screen-max-freq* | float | `999.0` | Screen out detections with frequencies greater than this value (Hz). |
| *det-output-beam-max-slow* | float | `1.667` | Write detection display-beams for detections with measured slowness less than this value (s/km). |
| *det-max-snr* | float | `50.0` | Use this value in the *deltime* computation if the measured snr exceeds this value. |
| *det-min-snr* | float | `5.0` | Use this value in the *deltime* computation if the measured snr is below this value. |

**TABLE 12: DETECTION PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *det-max-deltime* | float | `1.0` | Use this value if the computed *deltime* exceeds this value (s). |
| *det-min-deltime* | float | `0.1` | Use this value if the computed deltime is below this value (s). |

### Fk Recipe

The fk recipe parameters control the frequency wavenumber (f-k) computation (see Table 13). F-k analysis determines the direction and coherency of an incoming signal. The par files are organized by station and are located in the directory `…config/app_config/automatic/DFX/fk`. This processing applies to array stations only.

**TABLE 13: FK PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *fk-chan* | string | `"sz"` | Array element component to use. |
| *fk-bandwidth* | float | `2.0` | Bandwidth for f-k analysis (Hz). The analysis band is chosen based on either the *fk-bandwidth* and the arrival frequency, or on the *fk-fklof* and *fk-fkhif* values. If a valid *fk-bandwidth* value (non-zero) and valid arrival frequency are provided, the analysis band is centered on the detection frequency. Otherwise, the analysis band is defined by *fk-fklof* and *fk-fkhif*. |
| *fk-lead* | float | `1.1` | Time before the detection to begin the processing interval (s). |
| *fk-lag* | float | `4.9` | Time after the detection to end the processing interval (s). |
| *fk-max-slow* | float | `0.35` | Maximum slowness extent of the f-k analysis grid (s/km). |

**TABLE 13: FK PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *fk-min-sta* | int | `4` | Minimum number of array elements allowable for processing. |
| *fk-nslow* | int | `41` | Grid dimension. The grid will be an *fk-nslow* × *fk-nslow* grid. This value must be odd. |
| *fk-fklof* | float | `-1.0` | Lower limit of the analysis band (Hz). See *fk-bandwidth*. |
| *fk-fkhif* | float | `-1.0` | Upper limit of the analysis band (Hz). See *fk-bandwidth*. |
| *fk-power-output* | int | `1` | Flag indicating the type of output for the f-k grid; `0`: fstat, `1`: power |
| *fk-filter* | int | `1` | Prefilter flag; `0`: do not prefilter, `1`: prefilter (see *fk-flo*, *fk-fhi*, *fk-order*, *fk-zp*, and *fk-ftype*). |
| *fk-filter-onset* | float | `30.0` | Time interval added before and after the processing interval to allow filtering to stabilize (s). See *beam-filter-onset-len* in "Table 7: Common Beam Processing Parameters" on page 45. |
| *fk-flo* | float | `0.5` | Prefilter low cut-off frequency (Hz). |
| *fk-fhi* | float | `5.0` | Prefilter high cut-off frequency (Hz). |
| *fk-forder* | int | `2` | Prefilter order. |
| *fk-fzp* | int | `1` | Prefilter zero-phase flag; `0`: non zero-phase (causal), `1`: zero-phase (acausal). |
| *fk-ftype* | string | `"BP"` | Prefilter filter type (see *ftype* in "Table 8: Beam-recipe Table Parameters" on page 46). |
| *fk-dk* | float | `-1.0` | Measurement error parameter used to compute *delslo* ($km^{-1}$). |
| *fk-ds* | float | `-1.0` | Modelling error parameter used to compute *delslo* (s/km). |

TABLE 13: FK PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *fk-samprate-tol* | float | `0.01` | Maximum difference between the sample rates of the array elements (samples/s). Elements used for the analysis must have sample rates that agree within this tolerance. |
| *fk-include* | string | NULL | List of array elements to include in the processing (in addition to the elements with components matching the *fk-chan* parameter). The string has the following form: `"sta1/chan1 sta2/chan2 ..."` |
| *fk-exclude* | string | NULL | List of array elements to exclude from processing. See *fk-include*. |
| *fk-time-delay-file* | string | NULL | Time delay file to use with time-domain beamforming. This parameter is used for large-aperture array processing. |

### Hydro Recipe

The hydro recipe parameters control the hydroacoustic feature measurement computations. The par files are located in the `...config/app_config/automatic/DFX/hydro` directory and are organized by individual station par files (Table 14). This processing is applicable only to hydroacoustic stations. For more information on hydroacoustic feature extraction see [Lan97g].

TABLE 14: HYDROACOUSTIC PROCESSING PARAMETERS

| Parameter | Type | Default | Description |
|---|---|---|---|
| *hydro-sta* | string | NULL | Element station for processing. |
| *hydro-chan* | string | NULL | Element component for processing. |
| *hydro-flo* | float | `4.0` | Filter low frequency cut-off (Hz). |
| *hydro-fhi* | float | `8.0` | Filter high frequency cut-off (Hz). |

**TABLE 14: HYDROACOUSTIC PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *hydro-ford* | int | `3` | Filter order. |
| *hydro-fzp* | int | `0` | Filter zero phase flag; `0`: non zero-phase (causal), `1`: zero-phase (acausal) |
| *hydro-ftype* | string | `"BP"` | Filter type (see *ftype* in "Table 8: Beam-recipe Table Parameters" on page 46). |
| *hydro-fonset* | float | `10.0` | Time interval (s) added before and after the processing interval to allow filtering to stabilize. See *beam-filter-onset-len* in "Table 7: Common Beam Processing Parameters" on page 45. |
| *hydro-adjust-group-delay* | int | `0` | Flag indicating that the beam should be time-shifted to account for the filter group delay. |
| *hydro-noise-lead* | float | `25.0` | Time before the detection to start the noise analysis interval (s). |
| *hydro-noise-len* | float | `15.0` | Duration of noise analysis interval (s). |
| *hydro-signal-lead* | float | `10.0` | Time before the detection to start the signal analysis interval (s). |
| *hydro-signal-len* | float | `240.0` | Duration of signal analysis interval (s). |
| *hydro-taper* | float | `0.02` | Fraction of the data to taper (see *libwave(3)* man pages). |
| *hydro-max-interval* | float | `100.0` | Maximum time to search for the signal onset time relative to the start of the signal interval (s). |
| *hydro-onset-thresh* | float | `1.20` | The onset threshold level is set to the product of this value and the RMS noise level in the noise analysis window. |

**TABLE 14: HYDROACOUSTIC PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *hydro-term-thresh* | float | `1.75` | The termination threshold level is set to the product of this value and the RMS noise level in the noise analysis window. |
| *hydro-min-thresh* | float | `3.5` | The signal must reach the value of the product of the RMS noise and *hydro-min-thresh* before attempting identification of signal termination. |
| *hydro-onset-window* | float | `4.0` | Duration of the sliding window used to determine the onset (s). This window is similar to the short-term average sliding window used in *Detection Processing*. The average value in the window is compared with the onset threshold to determine the onset time. |
| *hydro-term-window* | float | `20.0` | Duration of the sliding window used to determine the signal termination (s). See *hydro-onset-window*. |
| *hydro-aic-order* | int | `5` | Order used by the AIC autoregressive model to estimate the onset time. See [Kvæ95] for more information. |
| *hydro-aic-window* | float | `25.0` | Interval, centered on the initial onset estimate, to refine the onset estimation (s). |
| *hydro-cep-lead* | float | `2.0` | Lead time relative to the mean arrival time of the cepstrum analysis interval (s) (see [Lan97g]). |
| *hydro-cep-len* | float | `10.0` | Length of the cepstrum analysis window (s). |
| *hydro-cep-pulse-delay-min* | float | `0.02` | Quefrency lower bound for the cepstral peak search (s). |
| *hydro-cep-pulse-delay-max* | float | `2.0` | Quefrency upper bound for the cepstral peak search (s). |

TABLE 14: HYDROACOUSTIC PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *hydro-cep-ft-taper* | float | `0.1` | Percentage of the spectral interval to taper (similar to the *hydro-taper* parameter, but applied to the frequency-series rather than the time-series). |
| *hydro-cep-box-width* | float | `0.3` | Width of the cepstral smoothing interval (s). |
| *hydro-cep-box-npass* | int | `3` | Number of passes in the smoothing. |
| *hydro-cep-nse-npass* | int | `3` | Number of passes in the NSE spectral detrender (see [Lan97g]). |
| *hydro-cep-nse-tpass3* | float | `4.0` | Spike threshold value in the NSE spectral detrender (see [Lan97g]). |
| *hydro-cep-nse-shift* | float | `100.0` | Data shift to apply before detrending, to assure positive-definite data (s) (see [Lan97g]). |
| *hydro-cep-nse-guard1* | float | `0.2` | Spectral guard band centered on the frequency sample being detrended (Hz) (see [Lan97g]). |
| *hydro-cep-nse-guard2* | float | `0.01` | Cepstral guard band centered on the frequency sample being detrended (Hz) (see [Lan97g]). |
| *hydro-cep-nse-aveband1* | float | `5.0` | Spectral averaging bandwidth to identify longer duration trends (Hz) (see [Lan97g]). |

**TABLE 14: HYDROACOUSTIC PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *hydro-cepnse-aveband2* | float | `0.01` | Spectral averaging bandwidth to identify short duration trends (Hz) (see [Lan97g]). |
| *hydro-make-prob-weight* | int | `1` | Probability weighted peak time flag. If `1`, an estimate that utilizes the noise level is made of the peak arrival time. For example, if the signal contains 2 peaks of about the same amplitude, a low noise level will set the peak time close to the larger peak and a high noise level will set the peak time between the peaks. |

### Infra Recipe

The infra recipe parameters control the infrasonic detection and feature measurement computations (Table 15). The par files are located in the directory `…config/app_config/automatic/DFX/infra` and are organized by individual station par files. This processing is applicable only to infrasonic stations. See the *libinfra(3)* manual section for details of the infrasonic detection and feature computation algorithms.

**TABLE 15: INFRASONIC PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *sta* | string | NULL | The station array name. |
| *filter-file-name-infra* | string | `"./Infra_FIR_BPF_Specification_Set.txt"` | Path to the FIR filter specification file. |

**TABLE 15: INFRASONIC PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *refinement* | string | NULL | Type of method to use during the refinement of the coherence peak position in the slowness plane. Values are `"native"`, `"fk"`, `"n_ccc"`, and `"none"`. `"native"` specifies use of the same grid-value method used during the coarse peak computation. `"fk"` specifies use of the standard f-k grid-value method, `"n_ccc"` uses normalized cross-correlation grid-values, and `"none"` indicates no refinement; simply interpolate the coarse grid. |
| *slowness-bins-to-refine* | int | `16` | Number of grid points to use during the refinement. The specified value is the number of grid points on either side of the initial slowness estimate. |
| *slowness-width-for-peak-interpolate* | float | `2.0` | Slowness range for the refinement grid (s/km) (see *slowness-bins-to-refine*). |
| *coherent-integration-time* | float | `60.0` | Interval to use for the cross-correlation window during *Detection Processing* and *Feature Extraction* (s). |
| *update-time-of-processing* | float | `30.0` | Amount of time to shift between each cross-correlation computation (s). For example, if cross-correlations of array channels are done at time T to estimate coherence (that is, a detection), the next set of cross-correlations will take place at T + 30 s, and so on. |
| *combination-criterion* | int | `1` | Type of data combination for computing the slowness grid values; `-2`: mean of the absolute value of the normalized cross-correlations (ncc), `-1`: censored (remove lowest and highest values) mean of the ncc, `0`: median of the ncc, `1`: mean of the ncc, and `2`: maximum of the ncc. |

**TABLE 15: INFRASONIC PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *scale-fs-flag* | int | `0` | Flag indicating that the channels used for the *fstat* calculations should be scaled. If `1`, then each channel will be normalized to a total power of 1.0 before beamforming and *fstat* computations. |
| *sta-vs-lta-fs-flag* | int | `0` | If *Detection Processing* is using *fstat-mode*, use STA/LTA slowness grid values. |
| *lta-fs-weight* | float | `0.016666667` | Weight used to compute the long-term average when *sta-vs-lta-fs-flag* is in force. See *libinfra(3)* man pages for more detail. |
| *offset-time* | float | `60.0` | Obsolete. |
| *azimuth-criterion* | float | `25.0` | Threshold used to match new arrivals with pre-existing database arrivals (deg). |
| *slowness-criterion* | float | `1.0` | Threshold used to match new arrivals with pre-existing database arrivals (s/km). Also used as the threshold to identify slowness peak cluster memberships. |
| *sta-time* | float | `5.0` | Short-term average window length used during STA/LTA coincidence detection (s). |
| *gap-time* | float | `10.0` | Time difference between the oldest sample in the STA window and the newest sample in the LTA window (s). |
| *lta-time* | float | `45.0` | Long-term average window length used during the STA/LTA coincidence detection (s). |
| *sta-lta-threshold* | float | `3.0` | STA/LTA threshold used during coincidence detection. |
| *coherent-threshold* | float | `1.5` | Fstat threshold used during coincidence detection. |

TABLE 15: INFRASONIC PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *xfk-dk* | float | `1.0` | Scale of the f-k measurement error (see *fk-dk* in "Table 13:Fk Processing Parameters" on page 54). |
| *xfk-ds* | float | `0.0` | F-k modelling error (see *fk-ds* in "Table 13:Fk Processing Parameters" on page 54). |
| *spctrl-est-window-type* | int | `-3` | Type of windowing to apply to the time-series before spectral estimation. Values are `-3`: Parzen, `-2`: squared-cosine, `-1`: Hamming, `0`: simple `"boxcar"`, `1`: Hanning, and `2`: cosine-bell. |
| *spctrl-slope* | float | `-6.0` | Spectral slope to use during the corner-frequency estimation (dB/octave). |

### Onset Recipe

The onset recipe parameters control the detection onset-time estimation (Table 16). The onset-time is the actual start of the signal in the time-series, as opposed to the time that the characteristic detection function (STA/LTA) exceeds the detection threshold. The par files are located in the following directory: `...config/app_config/automatic/DFX/ons`.

### TABLE 16: ONSET PROCESSING PARAMETERS

| Parameter | Type | Default | Description |
|---|---|---|---|
| *ons-min-sta-fraction* | float | `0.1` | Fraction of available beam elements to allow a measurement. See *det-min-sta-fraction* in "Table 12: Detection Processing Parameters" on page 50. |
| *ons-min-sta-len-fraction* | float | `0.9` | Fraction of the processing interval that *ons-min-sta-fraction* must meet or exceed to allow a measurement. See *det-mon-sta-len-fraction* in "Table 12: Detection Processing Parameters" on page 50. |
| *ons-method* | int | `0` | If non-zero, use a "farms" onset method. The "farms" algorithm is not documented. See the *DFX* source code to obtain more information. |
| *ons-ar-order* | int | `10` | Order of the auto-regressive model for the AIC onset-time estimation method. |
| *ons-max-time-adjust* | float | `2.0` | Maximum time allowed between the initial onset time and the revised onset time (s). |
| *ons-signal-lead* | float | `0.5` | Time before the initial onset to start the whitened signal beam interval (s). |
| *ons-signal-len* | float | `3.0` | Length of the whitened signal beam interval (s). |
| *ons-noise-lead* | float | `3.5` | Time before the initial onset to start the noise interval (s). The noise interval is used to model the noise for signal whitening. |
| *ons-noise-len* | float | `3.0` | Length of the noise interval (s). |
| *ons-aic-lead* | float | `3.5` | Used for the "farms" algorithm (s). See *ons-method*. |
| *ons-aic-len* | float | `6.0` | Used for the "farms" algorithm (s). See *ons-method*. |

**TABLE 16: ONSET PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *ons-aic-damp* | float | `0.1` | Used for the "farms" algorithm. See *ons-method*. |
| *ons-aic-decimate* | int | `0` | Used for the "farms" algorithm. See *ons-method*. |
| *ons-ar-signal-fraction* | float | `0.1` | Used for the "farms" algorithm. See *ons-method*. |
| *ons-SNR-filter* | int | `0` | If `1`, use a set of filters in a filter-bank parameter table. The filter giving the highest snr is used for the onset computation. |

### Polarization Recipe

The polarization recipe parameters control the 3-C particle-motion measurement computations (Table 17). Polarization recipe parameters are located in the …`config/app_config/DFX/polar` directory.

**TABLE 17: POLARIZATION PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *polar-sta-list* | string | NULL | List of array components to use for processing (for example: `"ARA01,ARA02,ARA03"`). |
| *polar-chan-list* | string | NULL | List of components (for example: `"sz,sn,se"`). |
| *polar-flo* | float | `1.0` | Filter low cut-off frequency (Hz). |
| *polar-fhi* | float | `3.0` | Filter high cut-off frequency (Hz). |
| *polar-ford* | int | `3` | Filter order. |
| *polar-zp* | int | `0` | Filter zero-phase flag; `0`: non zero-phase (causal), `1`: zero-phase (acausal). |

TABLE 17: POLARIZATION PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *polar-ftype* | string | "BP" | Filter type. See *ftype* in "Table 8: Beam-recipe Table Parameters" on page 46 for a list of types. |
| *polar-fonset* | float | 10.0 | Filter onset buffer length (s). See *beam-filter-onset-len* in "Table 7: Common Beam Processing Parameters" on page 45. |
| *polar-window* | float | 5.0 | Length of the analysis sliding window (s). |
| *polar-overlap-fraction* | float | 0.1 | Time overlap between each *polar-window* (s). For example, with the default values, the second sliding window would start 0.1 s before the end of the previous window. |
| *polar-noise-lead* | float | 5.1 | Time before the detection to start the noise analysis interval (s). |
| *polar-noise-len* | float | 5.0 | Length of the noise analysis interval (s). |
| *polar-signal-lead* | float | 0.1 | Time before the detection to start the signal analysis interval (s). |
| *polar-signal-len* | float | 5.0 | Length of the signal analysis interval (s). |
| *polar-taper* | float | 0.1 | Fraction of the data to taper (see *DFX* man pages). |
| *polar-thresh* | float | 1.0 | Not currently used. |
| *polar-pamp3c* | int | 0 | Not currently used. |
| *polar-dk* | float | -1.0 | Station-specific measurement error used to estimate the *delslo* value (when deriving slowness from rectilinear particle motion) ($km^{-1}$). |

TABLE 17: POLARIZATION PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *polar-ds* | float | `-1.0` | Station-specific modelling error used to estimate the delslo value (when deriving slowness from rectilinear particle motion) (s/km). |
| *polar-alpha* | float | `0.0` | Value used to convert from the rectilinear particle motion derived incidence angle to slowness. See the *DFX* man pages for details. |
| *polar-use-azi-slow* | int | `0` | If `1`, use the azimuth and slowness values derived from rectilinear motion as the detection features. Typically, the f-k feature values are used. |
| *polar-adjust-group-delay* | int | `0` | If `1`, adjust the filtered data to account for the filter group delay. |

### Precond Recipe

The precond recipe parameters provide a set of preconditioning FIR filters currently used only during hydroacoustic processing. The files are located in the `…config/app_config/DFX/precond` directory. The filter list is provided in a parameter table named *precondlist* with a single column (*tablename*) that provides the names of the parameter tables containing FIR filter coefficients. The FIR filter coefficient parameter tables contain a single column (*coef*). Hydroacoustic processing currently uses only the coefficient table named *invresp*, and this value is hard-coded in the hydroacoustic processing function *get_correction_coefs* in the `DFX/libsrc/libhydro` directory.

### QC Recipe

The QC recipe parameters control the data quality check and correction parameters (Table 18). The data QC processing identifies spikes, noise, and dropouts and makes their location available to subsequent processing. The par files are located in

the directory ...config/app_config/automatic/DFX/qc. Samples identified as "bad" are stored in a QC mask structure. Some of the parameters in Table 18 refer to this mask information.

**TABLE 18: QC PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *qc-max-mask-fraction* | float | 0.333 | Maximum fraction of the interval that can contain masked data samples before the entire interval is masked and omitted from processing. For example, in the default configuration, if more than 1/3 of the data are masked then the entire interval is masked. |
| *qc-type-multi-component* | string | "all" | QC extended processing type. When "all" is specified, then the single channel QC processing is applied to array components prior to array QC processing. This parameter is used only when processing array data, and if the *qc-apply-extended* flag is set. |
| *qc-min-multi-component* | int | 2 | Minimum number of channels required for array QC processing. |
| *qc-apply-extended* | int | 1 | Array QC processing flag. If this flag is off or the number of available channels is less than *qc-min-multi-component*, then single channel QC processing is applied. |
| *qc-demean* | int | 1 | If 1, remove the mean value of non-masked data from each sample after QC processing. |
| *qc-fix* | int | 1 | Method to use for fixing bad data segments; 0: bad data samples are set to zero-value, 1: single samples are cubic interpolated and segments up to *qc-gap-samples* are linearly interpolated. |

**TABLE 18: QC PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *qc-gap-samples* | int | 4 | Minimum number of consecutive bad samples that indicate a dropout. No attempt is made to fix bad intervals larger than this. |
| *qc-gap-taper-fraction* | float | 0.5 | Percentage of the data gap (see *qc-gap-samples*) to taper from the valid data sample values. |
| *qc-interval-samples* | int | 200 | Number of samples to use in a QC processing window. The window shifts along the data interval until the entire interval is processed. |
| *qc-interval-overlap-fraction* | float | 0.1 | Fraction of the *qc-interval-samples* window length to overlap. For example, with the default values, 1/10 of the window overlaps with the previous window. The window length is set by the *qc-interval-samples* parameter. |
| *qc-niter* | int | 1 | Number of QC processing iterations during multi-component processing. Using more iterations can identify lower-level spikes. For example, if there are populations of high- and low-level spikes, the high-level spikes can raise the average element maximum value and hide the low-level spikes. After the first iteration removes the high-level spikes, the second iteration may identify the low-level spikes. |
| *qc-single-trace-spike-thresh* | float | 10.0 | Spike threshold for single traces. Sample i, s(i) is considered a spike when this threshold is exceeded by the ratio [s(i) – s(i–1)]/[s(i–1) – s(i–2)]. |
| *qc-spike-thresh* | float | 3.0 | Amplitude threshold that a sample must exceed to be flagged as a spike. The amplitude is compared with a percentile of the other data samples (see *qc-spike-statistic-value*). |

TABLE 18: QC PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *qc-spike-statistic* | string | "percentile" | Type of spike comparison statistic. Values are "percentile" and "average". See *qc-spike-thresh*. |
| *qc-spike-statistic-value* | float | 70.0 | Percentile value to use when qc-spike-statistic="percentile". |
| *qc-spike-dataset* | string | "all" | Type of sample data measures to use to obtain a baseline value for spike comparison during multi-component QC. Values are "diff" for absolute differences between adjacent samples, "data" for the absolute data value, or "all" to use both measures. |
| *qc-spike-window-samples* | int | 20 | Number of samples in percentile window during single-component (1-C) spike identification. |

### Smult Recipe

The smult recipe parameters control the source multiplicity measurement computations (Tables 19 and 20). Recipe parameters are located in the …config/app_config/DFX/smult directory and are organized by station-specific par files (for example, ASAR-smult.par). The multiplicity computation, using cepstral peak characteristics, is made over regions of the signal spectra that have a reasonable snr, which is computed using the signal and noise spectral amplitudes.

TABLE 19: SMULT PROCESSING PARAMETERS

| Parameter | Type | Default | Description |
|---|---|---|---|
| *smult-array* | string | NULL | Station name of the array being processed (for example, "ASAR"). |
| *smult-sta-list* | string | NULL | List of array elements to use for the multiplicity measurements (for example, "AS01,AS02"). |

**TABLE 19: SMULT PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *smult-chan* | string | `"sz"` | Component to use for processing. |
| *smult-noise-lead* | float | `5.3` | Time before the detection to start the noise analysis interval (s). |
| *smult-signal-lead* | float | `0.3` | Time before the detection to start the signal analysis interval (s). |
| *smult-noise-duration* | float | `5.0` | Length of the noise analysis interval (s). |
| *smult-signal-duration* | float | `5.0` | Length of the signal analysis interval (s). |
| *smult-taper-type* | string | `"cosine"` | Type of taper to use on the analysis window. Values are `"cosine"`, `"parzen"`, `"hanning"`, `"hamming"`, `"blackman"`, and `"welch"`. |
| *smult-taper-length* | float | `0.5` | Percentage of the window to taper. |
| *smult-dfsmooth* | float | `0.75` | Percentage of the frequency band to use as a smoothing window while computing the smoothed spectrum. |
| *smult-threshdb* | float | `3.0` | Bands in the signal spectrum with an snr (computed by comparing the signal and noise spectral amplitudes) greater than *smult-threshdb* are used for processing. |
| *smult-min-frequency-interval* | float | `4.0` | Minimum allowable frequency interval (bandwidth) above *smult-threshdb* to use as a valid interval (Hz). |
| *smult-max-frequency-bands* | int | `20` | Maximum number of bands allowable. |
| *smult-max-frequency* | float | `19.6875` | Maximum frequency in the spectra to use for processing (Hz). |

**TABLE 19: SMULT PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *smult-min-frequency* | float | `1.875` | Minimum frequency in the spectra to use for processing (Hz). |
| *smult-num-trend-coefs* | int | `3` | Order of the polynomial to fit to the log-spectrum for detrending. |
| *smult-max-peaks* | int | `20` | Maximum number of quefrency peaks to identify in the cepstrum. |
| *smult-significant-peak-thresh* | float | `0.032` | Cepstral peak amplitude threshold that identifies a significant peak, for cepstral peaks that are consistent for all arrivals. |
| *smult-phase-peak-thresh* | float | `0.005` | Threshold for identifying peaks for a single arrival's cepstrum. |
| *smult-response-type* | string | `"theoretical"` | Not used; hardwired to `"theoretical"` in source code. |
| *smult-resp-units* | string | `"d"` | Instrument response correction units; valid values are `"d"` (displacement), `"v"` (velocity), and `"a"` (acceleration). See the *libresponse* man pages for more information. |

**TABLE 20: SMULT SCHEME-ONLY PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *smult-splp-min-delta* | float | (none) | Minimum allowable station-to-event distance (deg). |
| *smult-splp-max-delta* | float | (none) | Maximum allowable station-to-event distance (deg). |
| *smult-tf-max-depth* | float | (none) | Maximum allowable event depth (km). |

### Splp Recipe

The short-period/long-period recipe parameters control the splp energy ratio computations, which are a part of event characterization (Tables 21 and 22). Recipe parameters are located in the …`config/app_config/DFX/splp` directory.

**TABLE 21: SPLP PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *splp-chan-list* | string | NULL | A list of allowable channels for splp processing. For example: `"bz,bn,be,BHZ,BHN,BHE,bhz,bhn,bhe"`. |
| *splp-sp-flo* | float | `1.0` | Short-period filter low cut-off frequency (Hz). |
| *splp-sp-fhi* | float | `8.0` | Short-period filter high cut-off frequency (Hz). |
| *splp-lp-flo* | float | `0.033` | Long-period filter low cut-off frequency (Hz). |
| *splp-lp-fhi* | float | `0.125` | Long-period filter high cut-off frequency (Hz). |
| *splp-ford* | int | `3` | Filter order. |
| *splp-zp* | int | `1` | Filter zero-phase flag. 0: non zero-phase (causal), 1: zero-phase (acausal) |
| *splp-ftype* | string | `"BP"` | Filter type. See *ftype* in "Table 8: Beam-recipe Table Parameters" on page 46 for a list of types. |
| *splp-fonset* | float | `10.0` | Filter onset buffer length (s). See *beam-filter-onset-len* in "Table 7: Common Beam Processing Parameters" on page 45. |
| *splp-taper* | float | `0.1` | Fraction of the data to taper (see *libwave(3)* man pages). |

**TABLE 21: SPLP PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *splp-vsp1* | float | `8.2` | Group velocity to determine the beginning of the short-period analysis window (in terms of the station-to-event distance) (km/s). |
| *splp-vsp2* | float | `4.5` | Group velocity to determine the end of the short-period analysis window (in terms of the station-to-event distance) (km/s). |
| *splp-vlp1* | float | `4.5` | Group velocity to determine the beginning of the long-period analysis window (in terms of the station-to-event distance) (km/s). |
| *splp-vlp2* | float | `2.5` | Group velocity to determine the end of the long-period analysis window (in terms of the station-to-event distance) (km/s). |
| *splp-drop-thr* | int | `120` | Allowable number of bad samples for splp data QC (see *qc-gap-samples* in "Table 18: QC Processing Parameters" on page 68). |
| *splp-adjust-group-delay* | int | `0` | If `1`, adjust the filtered data to account for the filter group delay. |
| *splp-inst-edge-len* | float | `20.0` | Time interval at the start of the data to ignore when computing the energy in the processing interval (s). |

**TABLE 22: SPLP SCHEME-ONLY PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *evch-splp-min-delta* | float | (none) | Minimum allowable station-to-event distance (deg). |

TABLE 22: SPLP SCHEME-ONLY PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *evch-splp-max-delta* | float | (none) | Maximum allowable station-to-event distance (deg). |
| *evch-tf-max-depth* | float | (none) | Maximum allowable event depth (km). |

### TF Recipe

The time-frequency (TF) recipe parameters control the TF analysis, which is part of event characterization (Tables 23 and 24). TF analysis identifies features consistent with a ripple-fire source by analyzing the frequency characteristics over time via a sonogram analysis (see [Jep98]). Recipe parameters are located in the …`config/app_config/DFX/tf` directory.

TABLE 23: TF PROCESSING PARAMETERS

| Parameter | Type | Default | Description |
|---|---|---|---|
| *tf-adv* | float | `10.0` | The length of the noise window (s). |
| *tf-tlen* | float | `40.0` | The total processing interval (signal + noise) length (s). |
| *tf-tsub* | float | `2.5` | The length of the sonogram interval window (s). |
| *tf-ps* | float | `20.0` | Offset between sonogram interval windows, in percentage of the sonogram interval windows. For example, with default values each window will shift by 20 percent of 2.5 s, or 0.5 s. |
| *tf-iin* | int | `1` | Time-series units: `0`: displacement, `1`: velocity, `2`: acceleration. |
| *tf-iout* | int | `1` | Output spectral amplitude units: `0`: displacement, `1`: velocity, `2`: acceleration. |

**TABLE 23: TF PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *tf-tbw* | float | `4.0` | Time-bandwidth product (number of Rayleigh bins) for use in the Thomson multi-taper procedure. See [Sle78], [Tho82]. |
| *tf-nfb* | int | `11` | Number of frequencies spanned by the wide-band filter. |
| *tf-nfn* | int | `5` | Number of frequencies spanned by the narrow-band filter. |
| *tf-f0* | float | `0.0` | Minimum frequency (Hz). |
| *tf-bandtype1* | string | `"-"` | Not currently used. |
| *tf-bandtype3* | string | `"-"` | Not currently used. |
| *tf-vert-chan* | string | `"-"` | Vertical component identifier (for example, `"bz"`). This parameter is set in the station specification par file. |
| *tf-3c-chans* | string | `"-"` | Components to use for processing (for example, `"bz,bn,be"`). This parameter is set in the station specification par file. |
| *tf-statype* | string | `"-"` | Station type. Values are `"3c"` for 3-C stations and `"1c"` for 1-C stations. Only 3-C station data are currently processed. |
| *tf-nettype* | string | `"-"` | Not currently used. |

**TABLE 24: TF SCHEME-ONLY PROCESSING PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *evch-tf-min-delta* | float | (none) | Minimum allowable station-to-event distance (deg). |
| *evch-tf-max-delta* | float | (none) | Maximum allowable station-to-event distance (deg). |

**TABLE 24: TF SCHEME-ONLY PROCESSING PARAMETERS (CONTINUED)**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *evch-tf-max-depth* | float | (none) | Maximum allowable event depth (km). |
| *evch-tf-phase-list* | string | (none) | List of phases to use for TF processing (for example, `"Pn,Pg"`). |

### TI Recipe

The time-interval (TI) recipe parameters provide time-interval information to processes such as beamforming and feature measurements (Table 25). Time intervals can be based on an absolute time (for example, an arrival time), or a theoretical phase travel time based on station-to-event distance. The time-interval definitions are stored in a parameter table, are keyed by *name*, and may be referenced by other parameters or recipes (for example, *tirec* in "Table 4: Amprec Recipe Table Parameters" on page 41). Recipe parameters are located in the `...config/app_config/DFX/ti` directory.

**TABLE 25: TI RECIPE TABLE PARAMETERS**

| Parameter | Type | Default | Description |
|---|---|---|---|
| *name* | string | NULL | Time-interval recipe identifying string (for example, `"lag4"`, `"dbeam"`). |
| *rmin* | float | `0.0` | Minimum distance allowable for this recipe (deg). Used only for distance type time-intervals. |
| *rmax* | float | `360.0` | Maximum distance allowable for this recipe (deg). Used only for distance type time-intervals. |
| *sphase* | string | `"-"` | Phase on which to base the interval start time. |
| *sgv* | float | `-1.0` | Group velocity on which to base the interval start time (km/s). |

TABLE 25: TI RECIPE TABLE PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *lead* | float | `0.0` | Number of seconds before the absolute or computed time to start the interval (for example, `sphase="P"`, `lead=5.0` starts the interval 5.0 s before the theoretical *P*-phase arrival) (s). |
| *ephase* | string | `"-"` | Phase on which to base the interval end time. |
| *egv* | float | `-1.0` | Group velocity on which to base the interval end time (km/s). |
| *lag* | float | `0.0` | Number of seconds after the absolute or computed time to end the interval (for example, `sphase="P"`, `lag=10.0` ends the interval 10.0 s after the theoretical *P*-phase arrival) (s). |

### TM Recipe

The threshold monitoring (TM) recipe parameters provide parameters to control the threshold monitoring beam computation and output (Table 26). The threshold monitoring beam data are comprised of short-term average values. Recipe parameters are located in the …`config/app_config/DFX/tm` directory.

TABLE 26: TM PROCESSING PARAMETERS

| Parameter | Type | Default | Description |
|---|---|---|---|
| *tm-output-directory* | string | NULL | Path to the top-level directory containing the threshold-monitoring circular beam files. |
| *tm-null-val* | float | `-1.0` | Data value to use in missing data intervals. |
| *tm-stav-len* | float | `1.0` | Short-term average window length. |

TABLE 26: TM PROCESSING PARAMETERS (CONTINUED)

| Parameter | Type | Default | Description |
|---|---|---|---|
| *tm-stav-step* | float | `1.0` | Sample interval to use for the short-term average output beam. |
| *tm-cf-length* | float | `86400.0` | Number of samples in the circular file. |

### TMF Recipe

The third moment of frequency (TMF) recipe parameters provide parameters to control the TMF computation and output (see Table 27). The TMF value provides a measure of the amount of high-frequency energy contained in a signal, and is used as part of the event-characterization process. Recipe parameters are located in the …`config/app_config/DFX/tmf` directory. During the TMF computation most processing parameters are obtained from the smult configuration parameters. The TMF specific values replace the *smult-max-frequency* and *smult-min-frequency* values.

TABLE 27: TMF PROCESSING PARAMETERS

| Parameter | Type | Default | Description |
|---|---|---|---|
| *tmf-fmin* | float | required | Minimum frequency used in the processing band (Hz). |
| *tmf-fmax* | float | required | Maximum frequency used in the processing band (Hz). |

## Application Configuration Files

The following sections describe the organization and relationship of configuration files for each of the applications.

## Detection Processing

### Recipe File List and Interaction

*Detection Processing* (Seismic) is initiated by the *tuxshell* process, which provides the dynamic processing parameters *sta*, *start-time*, and *end-time*. All other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file `DFX-detection.par` and the site-specific configuration file `DFX-site-detection.par`. Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file (…`config/station_specs/`<*STA*>`.par`). These characteristics, in turn, are used by the site-specific configuration file `DFX-site-detection.par` to load the appropriate recipes. For example, `DFX-site-detection.par` loads the array QC recipe when the station is an array station, and the single-station QC recipe otherwise.

The following par files and recipes are used during *Detection Processing*:

- `DFX-detection.par`
- `DFX-site-detection.par`
- QC recipe (see "Table 18: QC Processing Parameters" on page 68.)
- Detection recipe (see "Table 12: Detection Processing Parameters" on page 50.)
- Onset recipe (see "Table 16: Onset Processing Parameters" on page 64.)
- Beam recipe (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- Fk recipe (see "Table 13: Fk Processing Parameters" on page 54.)
- Amp recipe (see "Table 3: General Amp Processing Parameters" on page 39 and "Table 4: Amprec Recipe Table Parameters" on page 41.)
- Amp3c recipe (see "Table 5: General Amp3c Processing Parameters" on page 42 and "Table 6: Amp3c Filter Band Table Parameters" on page 42.)

- Polar recipe (see "Table 17: Polarization Processing Parameters" on page 65.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)
- TM recipe (see "Table 26: TM Processing Parameters" on page 78.)

### Non-recipe Processing Parameters

Refer to [IDC7.1.1] for a description of other seismic *Detection Processing* parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-detection.scm`

### Example Recipe Modifications

The examples in this section are for beam recipe, detection recipe, and qc recipe modifications. Other *Detection Processing* recipe modifications are similar to these examples.

Recipe files are listed in `DFX-site-detection.par`. The recipe file name is typically constructed by using the *sta*, *NetType*, *WaveType*, and *StaType* parameters. The `sta` parameter is usually provided at runtime, and the latter three are defined in a station's station-specification file, located in the `…config/station_specs` directory.

### Modifying Beam Recipes

Detection beam recipe files are located in the `…config/app_config/DFX/beam/detection` directory. Files are named <*sta*>`-beam.par`. For example, the detection beam recipe file for station ASAR is in the file `ASAR-beam.par`. To remove a beam recipe, locate the recipe in the beam-recipe parameter table and

delete the line. To modify an existing recipe, locate the recipe and modify the appropriate column. To add a recipe, insert a line in the appropriate section of the recipe table; take care to ensure that the beam name field is unique.

### Modifying Detection Recipes

Detection recipe files are located in the …`config/app_config/DFX/det` directory. Files are named <*WaveType*>`det.par`. The value of *WaveType* for seismic processing is the empty string (for historical reasons) so, for example, the detection recipe file for the station ASAR is `det.par`. (Hydroacoustic stations use `hydro-` as the *WaveType*, so the hydro recipe file is `hydro-det.rec`). As all seismic stations share the same detection par file, any modifications to `det.par` affect all seismic station processing. To modify the recipe, edit the file `det.par`. Parameters not specified in the `det.par` file have default values for processing. Therefore, it may be necessary to add parameters and values to the file to set non-default values.

### Modifying QC Recipes

QC recipe files are located in the …`config/app_config/DFX/qc` directory. The name of the specific recipe file used is based on the *WaveType* and *NetType* values set in the station specification par file (Table 28; …`config/station_specs/`<*STA*>`.par`).

**TABLE 28: QC RECIPE FILE SELECTION**

| WaveType | NetType | QC Recipe File |
|---|---|---|
| `""` (seismic) | `"array"` | `array-qc.par` |
| `""` (seismic) | `"ss"` | `ss-qc.par` |
| `"hydro-"` | `"ss"` | `hydro-qc.par` |
| `"l"` (long-period) | `"array"` | `larray-qc.par` |

For example, the QC recipe file for station ASAR is `array-qc.par`. As all short-period array stations share the same QC par file, any modifications to `array-qc.par` affects all stations of this type. To modify the recipe, edit the file `array-qc.par`. Parameters not specified in the file use default value for processing, so it may be necessary to add parameters and values to the file.

### Recipe Files

Recipe files are loaded in the `DFX-site-detection.par` file, but it can be tricky finding them. It is best to start at the end of the file where a series of lines are found in the following format:

```
par=$(dfx-par-dir)/...
```

As an example, we will analyze the following line to see how the variables are set for the appropriate recipe:

```
par=$(dfx-par-dir)/qc/$(dfx-qc-recipe)
```

This line loads the QC recipe. The first variable `$(dfx-par-dir)` is set in the main processing file `DFX-detection.par` to the root of the *DFX* configuration tree, and it is in the form of *<rootpath>*`/config/app_config/DFX`. Thus, the file to be loaded is *<rootpath>*`/config/app_config/DFX/qc/$(dfx-qc-recipe)`. To decode the value of the `$(dfx-qc-recipe)` variable, move up the `DFX-site-detection.par` file to find the line:

```
dfx-qc-recipe=$(dfx-$(WaveType)$(NetType)qc)
```

Decoding this line from the inside out, we need to find the value of `dfx-$(WaveType)$(NetType)qc`. We will use station ASAR in this example and look in the station specification file for the values of the `WaveType` and `NetType` parameters. The station specification file `...config/station_specs/ASAR.par` has no entry for `WaveType` and the string `"array"` for `NetType`. The string `dfx-$(WaveType)$(NetType)qc` is thus `dfx-arrayqc`, so `dfx-qc-recipe` is set to the value of the variable `$(dfx-arrayqc)`. Looking further up the in the file

DFX-site-detection.par, we see the line dfx-arrayqc=array-qc.par. Thus, $(dfx-qc-recipe)=array-qc.par and the file loaded in the original line par=$(dfx-par-dir)/qc/$(dfx-qc-recipe) is:

par=<*rootpath*>/config/app_config/DFX/qc/array-qc.par

### Adding a Station

Many of the recipe files are not station specific. By setting the proper parameters in the station-specification file the correct recipes are loaded. The following is a list of the files that you must create when adding a new station for seismic *Detection Processing*. In this example the new station has the name QUUX, which is a 3-C array station.

- …config/station_specs/QUUX.par
  Include the parameters NetType=array, and Statype=3c.
- …config/app_config/DFX/beam/QUUX-beam.par
  This file contains the beam element group information for the elements in the QUUX array. See "Beam Element Groups" on page 48 and an existing 3-C array station element group par file as a guide for creating QUUX-beam.par.
- …config/app_config/DFX/beam/detection/QUUX-beam.par
  This file contains the beam recipe table. See "Beam Recipe" on page 43 and an existing 3-C array recipe file as a guide. Include the line par=$(PARDIR)/beam/$(sta)-beam.par at the top of the file to include the beam element group information.
- …config/app_config/DFX/fk/QUUX-fk.par
  This file contains f-k recipe information. See "Fk Recipe" on page 54 for a description of the recipe parameters. Use the values in the *XfkDisplay* configuration files if they exist for the new station. Otherwise, use parameter values from the configuration file of a similar station as an initial configuration.

- …config/app_config/DFX/polar/QUUX-polar.par
  This file contains the particle-motion analysis parameters. See "Polarization Recipe" on page 65 for a description of the recipe parameters. If no station-specific parameters are available, use the values of a similar station as an initial configuration.

#### Log File Location

If problems arise during detection processing, clues about the problem cause will likely be available in the program output. This output is stored in the file: *<LOGDIR>/<JDATE>/DFX/<STA>* where *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002), and *<STA>* is the station being processed. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>/<JDATE>/DFX-<HOST>-<PID>* where *<HOST>* is the name of the machine running the *DFX* instance and *<PID>* is the process ID.

### Origin Beam

#### Recipe File List and Interaction

*Origin Beam* is initiated by the *tuxshell* process as configured by the file …config/app_config/distributed/tuxshell/sel3/tuxshell-Beamer.par. *Tuxshell* provides the dynamic processing parameters *net*, *start-time*, and *end-time*. Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file DFX-originbeam.par and the site-specific configuration file DFX-site-originbeam.par. Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file …config/station_specs/*<STA>*.par. These characteristics, in turn, are used by the site-specific configuration file DFX-site-originbeam.par to load the appropriate recipes.

The following par files and recipes are used during *Origin Beam* processing:

- DFX-originbeam.par

- DFX-site-originbeam.par
- QC recipe (see "Table 18: QC Processing Parameters" on page 68.)
- Beam recipe (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Origin Beam* processing parameters.

### Application Scheme File

The Scheme file for this application is: DFX-originbeam.scm

### Example Recipe Modifications

Please see the "Example Recipe Modifications" on page 86 under Beam-on-the-Fly.

### Adding a Station

Please see the "Adding a Station" on page 86 under Beam-on-the-Fly.

### Log File Location

If problems arise during *Origin Beam* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*/Beamer/Beamer-*<HOST>*-*<PID>*. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*/*<JDATE>*/tuxshell/Beamer-*<HOST>*-*<PID>*, where *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002), *<HOST>* is the name of the machine running the *Beamer* instance, and *<PID>* is the process ID.

## Beam-on-the-Fly

### Recipe File List and Interaction

*Beam-on-the-Fly* is initiated by the *tuxshell* process as configured by the file …`config/app_config/distributed/tuxshell/interactive/tuxshell-BOTF.par`, which is run in response to an IPC message from the *ARS* application. *Tuxshell* provides a number of dynamic processing parameters (see [IDC7.1.1] for details). Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file `DFX-botf.par` and the site-specific configuration file `DFX-site-originbeam.par`. Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file located at …`config/station_specs/`*<STA>*`.par`. These characteristics, in turn, are used by the site-specific configuration file `DFX-site-originbeam.par` to load the appropriate recipes.

The following par files and recipes are used during *Beam-on-the-Fly* processing:

- `DFX-botf.par`
- `DFX-site-originbeam.par`
- QC recipe (see "Table 18: QC Processing Parameters" on page 68.)
- Beam recipe (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)

### Application Scheme File

The Scheme file for this application is: `DFX-botf.scm`.

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Beam-on-the-Fly* processing parameters.

### Example Recipe Modifications

The examples in this section are for beam recipe and TI recipe modifications. Further examples of recipe modifications are given in other sections of this document.

Recipe files are listed in `DFX-site-originbeam.par`. The recipe file name is typically constructed by using the *sta*, *NetType*, *WaveType*, and *StaType* parameters, but in this example, only the `sta` parameter is relevant.

#### Modifying Beam Recipes

The origin beam recipe files are located in the `…config/app_config/DFX/beam/originbeam` directory. Files are named <*sta*>`-beam.par`. For example, the detection beam recipe file for the station ASAR is in the file `ASAR-beam.par`. To remove a beam recipe, locate the recipe in the beam-recipe parameter table and delete the line. To modify an existing recipe, locate the recipe and modify the appropriate column. To add a recipe, insert a line in an appropriate section of the recipe table (the order of recipes is not critical, though it is useful to group them logically; for example, detection beams, depth-phase beams, and so forth).

#### Modifying TI Recipes

The TI recipes provide time interval information. If you wish to change the beam duration, this is the file to modify. The TI recipe files are located in the `…config/app_config/DFX/ti` directory. The file `originbeam-ti.par` is used by the *Beam-on-the-Fly* process. Other origin-beam processes also use `originbeam-ti.par`. The recipes used by *Beam-on-the-Fly* are specified by the parameters *botf-tele-tirec-list* and *botf-reg-tirec-list* (for teleseismic- and regional-distance beams, respectively) located in the file `DFX-botf.par`. Locate the recipe you wish to modify in the *tirec* parameter table and change the values of the appropriate columns.

### Adding a Station

The following files must be created when adding a new station for *Beam-on-the-Fly* processing. In this example the new station has the name QUUX, which is a 3-C array station.

- …`config/station_specs/QUUX.par`
  Include the parameters `NetType=array`, and `Statype=3c`.
- …`config/app_config/DFX/beam/QUUX-beam.par`
  This file contains the beam element group information for the elements in the QUUX array. See "Beam Element Groups" on page 48 and an existing 3-C array station element group par file as a guide for creating `QUUX-beam.par`.
- …`config/app_config/DFX/beam/originbeam/QUUX-beam.par`
  This file contains the beam recipe table. See "Beam Recipe" on page 43 and an existing 3-C array recipe file as a guide. The recipe set must include entries specified by the parameters *botf-tele-beamrec-list* and *botf-reg-beamrec-list* (located in the file `DFX-botf.par`). Include the line `par=$(PARDIR)/beam/$(sta)-beam.par` at the top of the file to load in the beam element group information.

### Log File Location

If problems arise during *Beam-on-the-Fly* processing, clues about the problem cause will likely be in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*`/interactive/Beamer-`*<HOST>*`-`*<PID>* where *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, `2002069` for the 69th day of 2002), *<HOST>* is the name of the machine running the *Beamer* instance, and *<PID>* is the process ID. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*/*<JDATE>*`/interactive/tuxshell-Beamer-`*<HOST>*`-`*<PID>*.

## Depth-phase SNR

### Recipe File List and Interaction

*Depth-phase SNR* is initiated by the *tuxshell* process as configured by the file …`config/app_config/distributed/tuxshell/recall/tuxshell-DFX-depth-phase-snr.par` and is part of the *recall* pipeline. *Tuxshell* provides the dynamic processing parameters *start-time*, *end-time*, and *net*. Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file `DFX-depth-phase-snr.par`, `DFX-recall.par`, and the site-specific configuration file `DFX-site-detection.par` (see [IDC7.1.1] for run-time parameter details). Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file located at …`config/station_specs/<STA>.par`. These characteristics, in turn, are used by the site-specific configuration file `DFX-site-detection.par` to load the appropriate recipes. For example, `DFX-site-detection.par` loads the array QC recipe when the station is an array station, and the single-station QC recipe otherwise. Although *Depth-phase SNR* processing uses the same site-specific configuration file as *Detection Processing*, many of the recipes are ignored (for example, fk recipe, polarization recipe) due to the limited scope of *Depth-phase SNR* processing.

The following par files and recipes are used during *Detection Processing*:

- `DFX-depth-phase-snr.par`
- `DFX-recall.par`
- `DFX-site-detection.par`
- QC recipe (see "Table 18: QC Processing Parameters" on page 68.)
- Detection recipe (see "Table 12: Detection Processing Parameters" on page 50.)
- Beam recipe (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)

- Amp recipe (see "Table 3: General Amp Processing Parameters" on page 39 and "Table 4: Amprec Recipe Table Parameters" on page 41.)

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Depth-phase SNR* processing parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-depth-phase-snr.scm`

### Example Recipe Modifications

The examples in this section are for beam and amplitude recipe modifications. Further examples of recipe modifications are given in other sections of this document.

Recipe files are listed in `DFX-site-detection.par`. The recipe file name is typically constructed by using the *sta*, *NetType*, *WaveType*, and *StaType* parameters, but in this example, only the `sta` parameter is relevant.

### Modifying Beam Recipes

The *Depth-phase SNR* beams are located in the detection beam recipes in the `…config/app_config/DFX/beam/detection` directory. Files are named <*sta*>`-beam.par`. For example, the detection beam recipe file for the station ASAR is in the file ASAR`-beam.par`. The recipe table section containing the *Depth-phase SNR* entries are located below the detection beams and are indicated by comments. The *Depth-phase SNR* beam recipe set is obtained from the amplitude recipes *beamrec* attributes. (The amplitude recipe set, in turn, is obtained through the `DFX-recall.par` parameter *depth-phase-snr-amprec-list*.)

To add an additional *Depth-phase SNR* measurement you must add a new beam recipe with the desired filter attributes. Use a unique beam *name* attribute. You must also add an amplitude recipe referencing this beam in the file `amp.par` and reference the amplitude measurement in the *depth-phase-snr-amprec-list* parameter located in `DFX-recall.par`.

### Modifying Amp Recipes

To add an additional *Depth-phase SNR* measurement you must add the signal and noise amplitude measurement information to the amplitude recipe table and create a measurement beam recipe with the desired filter (see the previous section). The amplitude recipe is located in the file …`/config/app_config/DFX/amp/amp.par`. The *Depth-phase SNR* section is identified by comments explaining the proper recipe nomenclature. As an example, we will add a measurement for the 1.0–3.5 Hz band to `amp.par`. The new entries will look like:

```
Name    Phase tirec  beamrec mtype       mvalue iresp fresp
dpN1035 -     DPlead dp1035  peak_trough     -1     0     0
dpS1035 -     DPlag  dp1035  peak_trough     -1     0     0
```

The first entry is the noise measurement recipe, and the second is the signal recipe. The only difference between the two is the *tirec* attribute. Reviewing the ti recipe in the file …`config/app_config/DFX/amp/ti.par` we see that the *DPLead* interval extends 5.0 s before the detection and *DPLag* interval extends 5.0 s after the detection. Both recipes share the same measurement beam, `dp1035`, which should be set in the file …`config/app_config/DFX/beam/detection/ASAR-beam.par` and have a filter specified as 1.0–3.5 Hz. The following example shows the `ASAR-beam.par` entry:

```
Name   Type rot std snr azi slo phase flo fhi ford zp ftype group
dp1035 coh  no    0  -1  -1  -1 -     1.0 3.5    3  0 BP    vert
```

### Adding a Station

The following files must be created when adding a new station for *Depth-phase SNR* processing. In this example the new station has the name QUUX, which is a 3-C array station.

- …config/station_specs/QUUX.par
  Include the parameters NetType=array, and Statype=3c.
- …config/app_config/DFX/beam/QUUX-beam.par
  This file contains the beam element group information for the elements in the QUUX array. See "Beam Element Groups" on page 48 and an existing 3-C array station element group par file as a guide for creating QUUX-beam.par.
- …config/app_config/DFX/beam/detection/QUUX-beam.par
  This file contains the beam recipe table. See "Beam Recipe" on page 43 and an existing 3-C array recipe file as a guide. The recipe set must include entries specified by the *Depth-phase SNR* amplitude recipes. These are common to all stations and may be cut and pasted from an existing beam recipe for a similar station.

#### Log File Location

If problems arise during *Depth-phase SNR* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*/Recall/DFX-depth-phase-SNR. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*/*<JDATE>*/tuxshell/DFX-depth-phase-SNR-*<HOST>*-*<PID>*. *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002), *<HOST>* is the name of the machine running the *Depth-phase SNR* instance, and *<PID>* is the process ID.

### Noise Amplitude

#### Recipe File List and Interaction

*Noise Amplitude* is initiated by a *tuxshell* process, configured by the file …config/distributed/tuxshell/recall/tuxshell-DFX-noiseamp.par. *Tuxshell* provides the dynamic processing parameters *start-time*, *end-time*, and *net*. Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file DFX-noiseamp.par and the site-specific configura-

tion file `DFX-site-noiseamp.par` (see [IDC7.1.1] for run-time parameter details). Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file located at `…config/station_specs/<STA>.par`. These characteristics, in turn, are used by the site-specific configuration file `DFX-site-noiseamp.par` to load the appropriate recipes.

The following par files and recipes are used during *Noise Amplitude* processing:

- `DFX-noiseamp.par`
- `DFX-site-noiseamp.par`
- QC recipe (see "Table 18:QC Processing Parameters" on page 68.)
- Beam recipe (`beam/detection/<STA>-beam.par`) (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)
- Amp recipe (see "Table 3: General Amp Processing Parameters" on page 39 and "Table 4: Amprec Recipe Table Parameters" on page 41.)

### Non-recipe processing parameters

See [IDC7.1.1] for a description of other *Noise Amplitude* processing parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-noiseamp.scm`

### Example Recipe Modifications

See the examples in "Example Recipe Modifications" on page 81 of Detection Processing.

### Adding a Station

See the examples in "Adding a Station" on page 84 of Detection Processing.

### Log File Location

If problems arise during *Noise Amplitude* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*/recall/DFX-noiseamp where *<LOGDIR>* is set in the top-level *$IMSPAR* par file and *<JDATE>* is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002). The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*/*<JDATE>*/tuxshell/DFX-noiseamp-*<HOST>*-*<PID>* where *<HOST>* is the name of the machine running the *DFX* instance and *<PID>* is the process ID.

## Seismic Event Characterization

### Recipe File List and Interaction

*Seismic Event Characterization* is initiated by a *tuxshell* process, configured by the file …config/app_config/distributed/tuxshell/evch/tuxshell-DFX-evch.par. *Tuxshell* provides the dynamic processing parameters *start-time*, *end-time*, and *net*. Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file DFX-evch.par and the site-specific configuration file DFX-site-evch.par (see [IDC7.1.1] for run-time parameter details). Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file located at …config/station_specs/*<STA>*.par. These characteristics, in turn, are used by the site-specific configuration file DFX-site-evch.par to load the appropriate recipes. For example, DFX-site-evch.par loads the array QC recipe when the station is an array station and the single-station QC recipe otherwise.

The following par files and recipes are used during *Seismic Event Characterization* processing:

- DFX-evch.par

- DFX-site-evch.par
- QC recipe (see "Table 18: QC Processing Parameters" on page 68.)
- Beam recipe in …beam/evch/evch-beam.par (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)
- Amp recipe (see "Table 3: General Amp Processing Parameters" on page 39 and "Table 4: Amprec Recipe Table Parameters" on page 41.)
- Smult recipe (see "Table 19: Smult Processing Parameters" on page 70.)
- Splp recipe (see "Table 21: Splp Processing Parameters" on page 73.)
- Complexity recipe (see "Table 11: Complexity Processing Parameters" on page 49.)
- TF recipe (see "Table 23: TF Processing Parameters" on page 75.)
- TMF recipe (see "Table 27: TMF Processing Parameters" on page 79.)

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Seismic Event Characterization* processing parameters.

### Application Scheme File

The Scheme file for this application is: DFX-evch.scm

### Example Recipe Modifications

The examples in this section are for splp recipe and smult recipe modifications. Further examples of recipe modifications are given in other sections of this document.

Recipe files are listed in `DFX-site-evch.par`. The recipe file name may be constructed by using the *sta*, *NetType*, *BandType3*, and *StaType* parameters set in the station specification par file.

### Modifying Splp Recipes

The splp par files are located in the `…config/app_config/DFX/splp` directory. Splp processing is currently performed only on single-station (non-array) 3-C broad-band stations. To modify the splp parameters, edit the `splp.par` file.

### Modifying Smult Recipes

Smult par files are located in the `…config/app_config/DFX/smult` directory. Smult parameters are station-specific so, for example, the station QUUX parameters will be in the file `QUUX-smult.par`. To modify the smult parameters, edit the appropriate file and modify the parameters as desired.

### Adding a Station

Many of the recipe files are not station specific. By setting the proper parameters in the station-specification file, the correct recipes are loaded. The following is a list of the files you must create when adding a new station for *Seismic Event Characterization* processing. In this example the new station has the name QUUX, which is a 3-C short-period array station.

- `…config/station_specs/QUUX.par`
  Include the parameters `NetType=array`, `Statype=3c`, and `BandType3=s`.
- `…config/app_config/DFX/beam/QUUX-beam.par`
  This file contains the beam element group information for the elements in the QUUX array. See "Beam Element Groups" on page 48 and an existing 3-C array station element group par file as a guide for creating `QUUX-beam.par`. (The evch beam recipes are not station-specific so it is not necessary to add a QUUX beam recipe file.)

- …config/app_config/DFX/smult/QUUX-smult.par
  This file contains the smult recipe information. See "Smult Recipe" on page 70 for a description of the recipe parameters. Use parameter values from the configuration file of a similar station as an initial configuration.

#### Log File Location

If problems arise during *Seismic Event Characterization* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*/Evch/DFX-evch-*<HOST>*-*<PID>* where *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002), *<HOST>* is the name of the machine running the *DFX* instance, and *<PID>* is the process ID. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*/*<JDATE>*/tuxshell/DFX-evch-*<HOST>*-*<PID>*.

### Hydro Detection

#### Recipe File List and Interaction

*Hydro Detection* is initiated by a *tuxshell* process, configured by the file …config/app_config/distributed/tuxshell/detpro/tuxshell-DFX.par. *Tuxshell* provides the dynamic processing parameters *start-time*, *end-time*, and *sta*. Other processing parameters are obtained from parameter and recipe files listed in the top-level configuration file DFX-detection.par and the site-specific configuration file DFX-site-detection.par (see [IDC7.1.1] for run-time parameter details). Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file located at …config/station_specs/*<STA>*.par. These characteristics, in turn, are used by the site-specific configuration file DFX-site-detection.par to load the appropriate recipes.

The following par files and recipes are used during *Hydro Detection* processing:

- DFX-detection.par

- DFX-site-detection.par
- QC recipe hydro-qc.par (see "Table 18: QC Processing Parameters" on page 68.)
- Beam recipe (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- Detection recipe hydro-det.par (see "Table 12: Detection Processing Parameters" on page 50.)
- Onset recipe hydro-ons.par (see "Table 16: Onset Processing Parameters" on page 64.)
- TI recipe ti.par (see "Table 25: TI Recipe Table Parameters" on page 77.)
- Amp recipe amp.par (see "Table 3: General Amp Processing Parameters" on page 39 and "Table 4: Amprec Recipe Table Parameters" on page 41.)
- Hydro recipe (see "Table 14: Hydroacoustic Processing Parameters" on page 56.)
- TM recipe tm.par (see "Table 26: TM Processing Parameters" on page 78.)

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Hydro Detection* processing parameters.

### Application Scheme File

The Scheme file for this application is: DFX-detection.scm

### Example Recipe Modifications

The example in this section is for a hydro recipe modification. Other examples of recipe modifications relevant to *Hydro Detection* processing are given in "Detection Processing" on page 80 of this document.

The recipe file is listed in the file `DFX-site-detection.par`. The hydro recipe file name is constructed from the *sta* parameter provided at runtime as <*STA*>-`hydro.par`.

### Modifying Hydro Recipes

The `hydro` par files are located in the …`config/app_config/DFX/hydro` directory. To modify the `hydro` parameters edit the <*STA*>-`hydro.par` file.

### Adding a Station

For example purposes, we will call our new station WET, with the station component `sp`.

- Follow the instructions for "Adding a Station" on page 84 of the Detection Processing section.

  When adding the station-specification file, set the following parameters:

  ```
  WaveType=hydro-
  NetType=ss
  StaType=1c
  hydro-sta=WET
  hydro-chan=sp
  ```

  (See an existing hydroacoustic station file as an example.)
- Add the hydro configuration file `WET-hydro.par` to the directory …`config/app_config/DFX/hydro`. Use the values from a similar station as a starting point for the configuration values.

### Log File Location

If problems arise during *Hydro Detection* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*/DFX/*<STA>* where *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002), and *<STA>* is the station being processed. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*/*<JDATE>*/DFX-*<HOST>*-*<PID>* where *<HOST>* is the name of the machine running the *DFX* instance and *<PID>* is the process ID.

## Hydro Recall

*Hydro Recall* is initiated by *ARS*. The processing information is sent from *ARS* to a *tuxshell* process, configured by the file …config/app_config/distributed/tuxshell/interactive/tuxshell-RHYDRO.par. *Hydro Recall* uses the same configuration as "Hydro Detection" on page 98, with the exception of the Non-recipe Processing Parameters.

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Hydro Recall* processing parameters.

### Application Scheme File

The Scheme file for this application is: DFX-hydro-recall.scm

### Log File Location

If problems arise during *Hydro Recall* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*/interactive/DFXHydroRecall-*<HOST>*-*<PID>* where *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002),

<*HOST*> is the name of the machine running the *DFX* instance, and <*PID*> is the process ID. The *tuxshell* output also may provide helpful information and is located in the file <*LOGDIR*>/<*JDATE*>/`interactive/tuxshell-DFXHydroRecall-`<*HOST*>`-`<*PID*>.

## Hydro Event Characterization

### Recipe File List and Interaction

*Hydro Event Characterization* is initiated by the *tuxshell* process as configured by the file …`config/app_config/distributed/tuxshell/evch/tuxshell-DFX-evch-hydro.par`. *Tuxshell* provides the dynamic processing parameters *start-time*, *end-time*, and *net*. Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file `DFX-evch-hydro.par` and the site-specific configuration file `DFX-site-evch-hydro.par` (see [IDC7.1.1] for run-time parameter details). Like the other applications, station characteristics are obtained from the station's specification file located at …`config/station_specs/`<*STA*>`.par`. These characteristics, in turn, are used by the site-specific configuration file `DFX-site-evch-hydro.par` to load the appropriate recipes.

The following par files and recipes are used during *Hydro Event Characterization* processing:

- `DFX-evch-hydro.par`
- `DFX-site-evch-hydro.par`
- Hydro recipe <*STA*>`-hydro.par` (see "Table 14: Hydroacoustic Processing Parameters" on page 56.)
- QC recipe `hydro-qc.par` (see "Table 18: QC Processing Parameters" on page 68.)
- Beam recipe (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)

- Amp recipe (see "Table 3: General Amp Processing Parameters" on page 39, and "Table 4: Amprec Recipe Table Parameters" on page 41.)

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Hydro Event Characterization* processing parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-evch-hydro.scm`

### Example Recipe Modifications

All of the recipe files used by *Hydro Event Characterization* are shared by other hydro processes. See "Hydro Detection" on page 98 for recipe modification examples.

### Adding a Station

The same modifications for adding a *Hydro Event Characterization* station are used for *Hydro Detection*. See "Hydro Detection" on page 98 for a description.

### Log File Location

If problems arise during *Hydro Event Characterization* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*/*<JDATE>*/`Evch/DFX-evch-hydro-`*<HOST>*`-`*<PID>*. *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, `2002069` for the 69th day of 2002), *<HOST>* is the name of the machine running the *DFX* instance, and *<PID>* is the process ID. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*/*<JDATE>*/`tuxshell/DFX-evch-hydro-`*<HOST>*`-`*<PID>*.

## Infra Detection

### Recipe File List and Interaction

*Infra Detection* is initiated by a *tuxshell* process, configured by the file …`config/app_config/distributed/tuxshell/detpro/tuxshell-DFX.par`. *Tuxshell* provides the dynamic processing parameters *start-time*, *end-time*, and *sta*. Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file `DFX-infra-detection.par`. Unlike most other *DFX* applications, *Infra Detection* processing loads processing recipes from the top-level configuration file rather than using a site-specific configuration file.

The following par files and recipes are used during *Infra Detection* processing:

- `DFX-infra-detection.par`
- QC recipe `infra-qc.par` (see "Table 18: QC Processing Parameters" on page 68.)
- Beam recipe (`beam/detection/<STA>-beam.par`) (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe `infra-ti.par` (see "Table 25: TI Recipe Table Parameters" on page 77.)
- Amp recipe `<STA>-infra-amp.par` (see "Table 3: General Amp Processing Parameters" on page 39 and "Table 4: Amprec Recipe Table Parameters" on page 41.)

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Infra Detection* processing parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-infra-detection.scm`

### Example Recipe Modifications

The infrasonic par files are located in the …config/app_config/DFX/infra directory. To modify a station's infrasonic recipe parameters, edit the <*STA*>-infra.par file.

### Adding a Station

To add an infrasonic station:

- Follow the instructions in the "Adding a Station" on page 84 for adding a station.
- Add the station-specific infrasonic par file to the …config/app_config/DFX/infra directory as <*STA*>-infra.par.
- Add the station-specific infrasonic QC recipe file to the …config/app_config/DFX/qc directory, as <*STA*>-infra-qc.par.
- Add the station-specific infrasonic amplitude recipe file to the …config/app_config/DFX/amp directory, as <*STA*>-infra-amp.par.

### Log File Location

If problems arise during *Infra Detection* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file <*LOGDIR*>/<*JDATE*>/DFX/<*STA*> where <*LOGDIR*> is set in the top-level *$IMSPAR* par file, <*JDATE*> is the GMT Julian date when the data were processed (for example, 2002069 for the 69th day of 2002), and <*STA*> is the station being processed. The *tuxshell* output also may provide helpful information and is located in the file <*LOGDIR*>/<*JDATE*>/DFX-<*HOST*>-<*PID*> where <*HOST*> is the name of the machine running the *DFX* instance and <*PID*> is the process ID.

## Interactive Recall Processing

*Interactive Recall Processing* is the seismic recall process initiated by *ARS*. The processing information is sent from *ARS* to a *tuxshell* process, configured by the file `...config/app_config/distributed/tuxshell/interactive/tuxshell-RSEISMO.par`. *Interactive Recall Processing* uses the same configuration as "Detection Processing" on page 80 with the exception of the "Non-recipe Processing Parameters" on page 87.

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Interactive Recall Processing* parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-int-recall.scm`

### Log File Location

If problems arise during *Interactive Recall Processing*, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*`/`*<JDATE>*`/interactive/DFXRecall-`*<HOST>*`-`*<PID>* where *<LOGDIR>* is set in the top-level *$IMSPAR* par file, *<JDATE>* is the GMT Julian date when the data were processed (for example, `2002069` for the 69th day of 2002), *<HOST>* is the name of the machine running the *DFX* instance, and *<PID>* is the process ID. The *tuxshell* output also may provide helpful information and is located in the file *<LOGDIR>*`/`*<JDATE>*`/interactive/tuxshell-DFXRecall-`*<HOST>*`-`*<PID>*.

## QC Stats

*QC Stats* is an option that is executed within the *Seismic Detection*, *Hydro Detection*, and *Infra Detection* applications. *QC Stats* uses no configuration recipe information. Configuration parameters are described in the Quality Control Statistics section of [IDC7.1.1].

### Application Scheme File

The Scheme file for this application is: `DFX-qcstats.scm`

## Seismic Recall

*Seismic Recall* is an automatic process that follows interactive analyst review (not to be confused with "Interactive Recall Processing" on page 106). The processing is initiated by a *tuxshell* process as configured by the file `…config/app_config/distributed/tuxshell/recall/tuxshell-DFX-recall.par`. *Seismic Recall* uses the same configuration as "Detection Processing" on page 80 with the exception of the Non-recipe Processing Parameters.

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Seismic Recall* processing parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-recall.scm`

### Log File Location

If problems arise during *Seismic Recall* processing, clues about the problem cause will likely be available in the program output. This output is stored in the file *<LOGDIR>*`/`*<JDATE>*`/Recall/DFX-recall` where *<LOGDIR>* is set in the top-level *$IMSPAR* par file and *<JDATE>* is the GMT Julian date when the data were processed (for example, `2002069` for the 69th day of 2002). The *tuxshell* output

also may provide helpful information and is located in the file <*LOGDIR*>/<*JDATE*>/`tuxshell/DFX-recall-`<*HOST*>`-`<*PID*> where <*HOST*> is the name of the machine running the *DFX* instance and <*PID*> is the process ID.

## Segment Archiving

### Recipe File List and Interaction

*Segment Archiving* is an automatic process initiated by a *tuxshell* process as configured by the file …`config/app_config/distributed/tuxshell/segarch/tuxshell-DFX-segarch.par`. *Tuxshell* provides the dynamic processing parameters *start-time*, *end-time*, and *net*. Other processing parameters are obtained from parameter and recipe files loaded by the top-level configuration file `DFX-segarch.par` and the site-specific configuration file `DFX-site-originbeam.par`. Station characteristics (array or single-station, 1-C or 3-C, and so forth) are obtained from the station's specification file located at …`config/station_specs/`<*STA*>`.par`. These characteristics, in turn, are used by the site-specific configuration file `DFX-site-originbeam.par` to load the appropriate recipes.

The following par files and recipes are used during *Segment Archiving* processing:

- `DFX-segarch.par`
- `DFX-site-originbeam.par`
- QC recipe (see "Table 18: QC Processing Parameters" on page 68.)
- Beam recipe (see "Table 7: Common Beam Processing Parameters" on page 45, "Table 9: Beam-group Table Parameters" on page 48, and "Table 10: Beam Element Table Parameters" on page 49.)
- TI recipe (see "Table 25: TI Recipe Table Parameters" on page 77.)

### Non-recipe Processing Parameters

See [IDC7.1.1] for a description of other *Segment Archiving* processing parameters.

### Application Scheme File

The Scheme file for this application is: `DFX-segarch.scm`

### Example Recipe Modifications

See "Example Recipe Modifications" on page 88 under Beam-on-the-Fly.

### Adding a Station

See "Adding a Station" on page 89 under Beam-on-the-Fly.

# Chapter 3: Troubleshooting

This chapter describes how to identify and correct problems related to *DFX* and includes the following topics:

# Chapter 3: Troubleshooting

## MONITORING

Monitoring the state of the *DFX* applications is best done by reviewing both their processing output and their log files. If you are reading this section it is likely that an application is encountering some problem, which was identified by some sort of change in its output quality or quantity. The first step is to examine the application's log file. The log file locations are provided in "Chapter 2: Operational Procedures" on page 19. For example, the *Interactive Recall Processing* log file location is given in "Log File Location" on page 106.

It may be that no log file has been written for the problem application. In this case, check processes further up the processing chain described in "Normal Operational Invocations" on page 20.

## INTERPRETING ERROR MESSAGES

The error messages described in this section are printed to the application log file. Only error messages related to immediate program termination are described.

If an error is related to a database query that you expect should be returning records but is not, try copying the query from the log file to an interactive SQL session and experimenting with the various query constraint clauses to find out why no records are being returned.

## Detection Processing Error Messages

| | |
|---|---|
| Message: | `Instrument response file <FILE> not accessible!` |
| Description: | An instrument response file defined in the **instrument** database table is not accessible to the application. |
| Action: | Check the consistency of the **instrument** table *dir* and *dfile* attributes with the response file location. Check that the application has read permission for the response file. |

| | |
|---|---|
| Message: | `Error submitting arrivals to database.` |
| Description: | An error occurred while writing results to the database. A message describing the table on which the write attempt failed should also appear. |
| Action: | Check the database state, the existence of the table, and table permissions. |

| | |
|---|---|
| Message: | `Fatal error encountered while processing at time <PROCESSING TIME>` |
| Description: | This top-level error message is printed (along with a more descriptive message) when a major processing error occurs. |
| Action: | Review the associated error messages for the cause. |

| | |
|---|---|
| Message: | `Error: Cannot initialize <TYPE> recipes for station <STA>` |
| Description: | An error occurred reading a processing recipe. |
| Action: | Verify that all recipe configuration files are available and properly configured. Review other associated error messages to help identify the problem. |

---

Message: `Error: Cannot initialize filter bank for station <STA>`

Description: The *ons-SNR-filter* parameter is set, but the filter bank is empty.

Action: The filter bank functionality is currently not used in this application. However, when it is enabled and you receive this error, then check the filter bank configuration. See the C-code function `libsrc/libutil/gobj_filter.c::init_filter_recipe_cvar()`.

---

Message: `Error: Cannot initialize wfdatas for station <STA>`

Description: No sensor channel information was available for the station <*STA*>.

Action: Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**.

---

Message: `Error: Problem determining data interval`

Description: A problem was encountered determining the data interval to process. The data interval is computed using the requested start-times and end-times, and other parameters such as the *data-offset-len*, and the long-term-average length.

Action: Check that the interval parameters are consistent (for example, that the *start-time* is earlier than the *end-time*).

---

Message: `DFX-detection(read-detection-waveforms) Error: No copy of wfdata container can be made`

Description: A copy process failed. This error should only be reported when no **wfdisc** data are available. If there are no **wfdiscs** for the processing interval, processing can not occur.

Action: See action for "`Error: No wfdiscs found`."

---

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `Error: Cannot read waveform data` |
| Description: | An error occurred while reading the waveform data from the disk files. |
| Action: | Verify that the disk files exist for the requested waveforms and that read permissions are set to allow the process to access the files. |

| | |
|---|---|
| Message: | `Error: Problem determining processing interval` |
| Description: | The cause of this error is similar to the cause of the error message `Error: Problem determining data interval`. |
| Action: | Follow the actions for `Error: Problem determining data interval`. |

| | |
|---|---|
| Message: | `Error: Cannot QC waveform data` |
| Description: | An error occurred during data quality-control processing. The problem may be a result of inconsistent input data or in the low-level common library code. |
| Action: | Check that the input data are consistent. If inconsistent data are ruled out, contact the application maintainer. |

| | |
|---|---|
| Message: | `Error: Cannot initialize station <STA>` |
| Description: | An error occurred during station initialization. |
| Action: | Review the adjacent error message for the problem cause. |

| | |
|---|---|
| Message: | `Error: Cannot measure any detection features` |
| Description: | This error will only occur if no detections have been made. |
| Action: | No actions are needed to resolve this condition. |

### Origin Beam Error Messages

| | |
|---|---|
| Message: | `Fatal error encountered while processing at time <PROCESSING TIME>` |
| Description: | This top-level error message is printed (along with a more descriptive message) when a major processing error occurs. |
| Action: | Review the associated error messages for the cause. |

| | |
|---|---|
| Message: | `Error: No sites found` |
| Description: | No **site** records were found in the database for the given network and processing interval. |
| Action: | Verify that the site information is configured properly in the database. The **site** table must contain *sta* values corresponding to the *sta* values for **affiliation**.*net* and the **site**.*ondate* and *offdate* values must bracket the time interval being processed. |

| | |
|---|---|
| Message: | `Error: No sites initialized` |
| Description: | The function organizing station recipes failed for all sites. |
| Action: | Verify that the recipe configurations are correct and that the top-level directory parameters are set correctly. The top-level configuration file pointed to by the *$IMSPAR* environment variable and the *DFXSPECS* parameter should have the parameter *PARDIR* set to the correct *DFX* configuration root directory. |

| | |
|---|---|
| Message: | `Error: No net time interval computed` |
| Description: | The computation of the processing time interval failed. This problem may be caused by incorrect specification of the TI recipe (see "TI Recipe" on page 77), or by errors in the low-level *DFX* library code. |
| Action: | Review adjacent error messages for additional information. |

| | |
|---|---|
| Message: | `Error: No wfdatas found` |
| Description: | No sensor channel information was available from the database. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `Error: No origins found` |
| Description: | No origins were returned from the database query. |
| Action: | There may actually be no origins in the processing interval. If there are origins, check that the database is operational and that the correct table, set by the parameter *origin-table*, is being accessed. |

| | |
|---|---|
| Message: | `Error: Cannot initialize sites` |
| Description: | The function organizing station, site, and waveform data failed for all sites. |
| Action: | Verify that the database is operational. Verify that the recipe configurations are correct and that the top-level directory parameters are set correctly. Verify that the station beam recipes are correctly figured (see "Seismic Event Characterization" on page 23). The top-level configuration file pointed to by the *$IMSPAR* environment variable and the *DFXSPECS* parameter should have the parameter *PARDIR* set to the correct *DFX* configuration root directory. |

### Beam-on-the-Fly Error Messages

| | |
|---|---|
| Message: | `error submitting wfdiscs to the database` |
| Description: | *DFX* encountered a database error while writing beam records to the **wfdisc** database table. |
| Action: | Verify the existence of the **wfdisc** database table. Verify that the database is operational. |

| | |
|---|---|
| Message: | `No wfdatas found` <*QUERY STRING*> |
| Description: | No sensor channel information was available for the <*QUERY STRING*>. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error forming channel string for wfdisc query` |
| Description: | No valid channel names were returned from the query to obtain station sensor channel information. |
| Action: | Verify that the station being processed is configured properly in the database. |

| | |
|---|---|
| Message: | `Error forming station string for wfdisc query` |
| Description: | No valid station was provided as a runtime parameter. |
| Action: | Verify that a valid station is provided to the application through the parameter *sta*. |

| | |
|---|---|
| Message: | `No wfdiscs found` <*WFDISC QUERY*> |
| Description: | No **wfdisc** records were found given the query <*WFDISC QUERY*>. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `No time interval could be computed for orid <ORID> stations <STATION LIST>` |
| Description: | The beam time interval could not be computed for at least one station in the station set. |
| Action: | The time interval is computed based on the station-to-event distance and phase slowness. Check low-level error messages written by the *DFX* library code. These messages should provide more detail on the cause of the error. |

### Depth-phase SNR Error Messages

| | |
|---|---|
| Message: | `Fatal error encountered while processing at time <CURRENT TIME>.` |
| Description: | A large number of stations or arrivals encountered errors during processing. |
| Action: | Review the non-fatal messages that follow this message. |

| | |
|---|---|
| Message: | `Error: Cannot initialize <TYPE> recipe(s) for station <STA>` |
| Description: | A recipe was not available for the particular station. |
| Action: | Verify that all recipes are available to the process and that they are properly configured (see "Depth-phase SNR" on page 90 for a list of required recipes). |

| | |
|---|---|
| Message: | `Error: Cannot initialize wfdatas for station <STA>` |
| Description: | No sensor channel information was available for the station <*STA*>. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error: Problem determining data interval` |
| Description: | A problem was encountered determining the data interval to process. The data interval is computed using the requested start-times and end-times, and other parameters such as the *data-offset-len*, and the long-term-average length. |
| Action: | Check that the interval parameters are consistent (for example, that the *start-time* is earlier than the *end-time*). |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `Error: Cannot read waveform data` |
| Description: | An error occurred while reading the waveform data from the disk files. |
| Action: | Verify that the disk files exist for the requested waveforms and that read permissions are set to allow the process to access the files. |

| | |
|---|---|
| Message: | `Error: Problem determining processing interval` |
| Description: | The cause of this error is similar to the cause of the error message `Error: Problem determining data interval`. |
| Action: | Follow the actions for `Error: Problem determining data interval`. |

| | |
|---|---|
| Message: | `Error: Cannot QC waveform data` |
| Description: | An error occurred during data quality-control processing. The problem may be a result of inconsistent input data or in the low-level common library code. |
| Action: | Check that the input data are consistent. If inconsistent data are ruled out, contact the application maintainer. |

| | |
|---|---|
| Message: | `Error: Azimuth or Slowness is NULL! Detection` <*ARID*> `at station` <*STA*> `has azimuth =` <*AZ*>`, slowness =` <*SLOW*> |
| Description: | The input arrival has a null azimuth or slowness. Processing assumes that the input arrivals have azimuth and slowness values; therefore, this arrival cannot be processed. |
| Action: | No action is necessary. |

| | |
|---|---|
| Message: | `Error measuring depth phase snr for arrival` *<ARID>* |
| Description: | A problem occurred either during the depth phase snr computation, or during inserts of the results to the **amplitude** database table, or this arrival has already been processed with its results stored in the database. |
| Action: | Errors during computation may be a result of data quality-control-masked waveform data, in which case there are no appropriate actions. A problem with the database is likely to be seen in other applications. Verify that the database is operational, and that the **amplitude** table is correctly specified (via the *out-amplitude-table* parameter). |

### Noise Amplitude Error Messages

| | |
|---|---|
| Message: | `Fatal error encountered while processing at time` *<PROCESSING TIME>* |
| Description: | This top-level error message is printed (along with a more descriptive message) when a major processing error occurs. |
| Action: | Review the associated error messages for the cause. |

| | |
|---|---|
| Message: | `Error: No sites found` |
| Description: | No **site** records were found in the database for the given network and processing interval. |
| Action: | Verify that the site information is configured properly in the database. The **site** table must contain *sta* values corresponding to the *sta* values for **affiliation**.*net* and the **site**.*ondate* and *offdate* values must bracket the time interval being processed. |

| | |
|---|---|
| Message: | `Error: No sites initialized` |
| Description: | The function organizing station recipes failed for all sites. |
| Action: | Verify that the recipe configurations are correct and that the top-level directory parameters are set correctly. The top-level configuration file pointed to by the *$IMSPAR* environment variable and the *DFXSPECS* parameter should have the parameter *PARDIR* set to the correct *DFX* configuration root directory. |

| | |
|---|---|
| Message: | `No net time interval computed` |
| Description: | Computation of the processing time interval failed. This problem is caused either by incorrect specification of the TI recipe (see "TI Recipe" on page 77), or by errors in the low-level *DFX* library code. |
| Action: | Review adjacent error messages for additional information. |

| | |
|---|---|
| Message: | `Error: No wfdatas found` |
| Description: | No sensor channel information was available from the database. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `Warning: Cannot get time interval for processing` |
| Description: | An error occurred during the computation of the station-to-event travel time, which is required in the determination of the processing interval. |
| Action: | Review the associated error messages for more information on the cause of the error. |

| | |
|---|---|
| Message: | `Error: Cannot read waveform data` |
| Description: | An error occurred while reading the waveform data from the disk files. |
| Action: | Verify that the disk files exist for the requested waveforms and that read permissions are set to allow the process to access the files. |

| | |
|---|---|
| Message: | `Error: Cannot QC waveform data` |
| Description: | An error occurred during data quality-control processing. The problem may be a result of inconsistent input data or in the low-level common library code. |
| Action: | Check that the input data are consistent. If inconsistent data are ruled out, contact the application maintainer. |

## Seismic Event Characterization Error Messages

| | |
|---|---|
| Message: | `No associated arrivals found in database` |
| Description: | No arrivals associated with the origin set were found in the database. |
| Action: | Verify that the database is operational and that the table names are correctly set (using parameters *in-arrival-table*, *in-assoc-table*, *origin-table*, and *affiliation-table*). |

| | |
|---|---|
| Message: | `Error: No sites found` |
| Description: | No **site** records were found in the database for the given network and processing interval. |
| Action: | Verify that the site information is configured properly in the database. The **site** table must contain *sta* values corresponding to the *sta* values for **affiliation**.*net* and the **site**.*ondate* and *offdate* values must bracket the time interval being processed. |

| | |
|---|---|
| Message: | `Error: No sites initialized` |
| Description: | The function organizing station recipes failed for all sites. |
| Action: | Verify that the recipe configurations are correct and that the top-level directory parameters are set correctly. The top-level configuration file pointed to by the *$IMSPAR* environment variable and the *DFXSPECS* parameter should have the parameter *PARDIR* set to the correct *DFX* configuration root directory. |

| | |
|---|---|
| Message: | `No net time interval computed` |
| Description: | The computation of the processing time interval failed. This problem may be caused by incorrect specification of the TI recipe (see "TI Recipe" on page 77), or by errors in the lower-level *DFX* library code. |
| Action: | Review adjacent error messages for additional information. |

| | |
|---|---|
| Message: | `Error: No wfdatas found` |
| Description: | No sensor channel information was available from the database. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `No origins found for processing` |
| Description: | No origins were returned from the database query. |
| Action: | There may actually be no origins in the processing interval. If there are origins, check that the database is operational and that the correct table, set by the parameter *origin-table*, is being accessed. |

| | |
|---|---|
| Message: | `Error: Cannot initialize sites` |
| Description: | The function organizing station site and waveform data failed for all sites. |
| Action: | Verify that the database is operational. Verify that the recipe configurations are correct and that the top-level directory parameters are set correctly. Verify that the station beam recipes are correctly figured (see "Seismic Event Characterization" on page 95). The top-level configuration file pointed to by the *$IMSPAR* environment variable and the *DFXSPECS* parameter should have the parameter *PARDIR* set to the correct *DFX* configuration root directory. |

| | |
|---|---|
| Message: | `Error: no origerr entry found for orid` <*ORID*> |
| Description: | An *orid* being processed has no corresponding **origerr** record. |
| Action: | Verify that the database is consistent (that the origin actually has an **origerr**). If it does, verify that the correct **origin** and **origerr** tables are being accessed via the parameters *origin-table* and *origerr-table*. |

### Hydro Detection Error Messages

| | |
|---|---|
| Message: | `Error in` <*DATABASE TABLE*> `submit` |
| Description: | An error occurred while writing results to a required database output table. |
| Action: | Verify that the database is operational. Verify that the target tables exist and have insert permission. |

| | |
|---|---|
| Message: | `Error: Cannot initialize recipes for station <STA>` |
| Description: | An error occurred reading a processing recipe. |
| Action: | Verify that all recipe configuration files are available and properly configured. Review other associated error messages to help identify the problem. |

| | |
|---|---|
| Message: | `Error: Cannot initialize wfdatas for station <STA>` |
| Description: | No sensor channel information was available for the station <*STA*>. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error: Problem determining data interval` |
| Description: | A problem was encountered determining the data interval to process. The data interval is computed using the requested start-times and end-times, and other parameters such as the *data-offset-len*, and the long-term-average length. |
| Action: | Check that the interval parameters are consistent (for example, that the *start-time* is earlier than the *end-time*). |

| | |
|---|---|
| Message: | `DFX-hydro-detection(read-hydro-detection-waveforms)`<br>`Error: No copy of wfdata container can be made` |
| Description: | A copy process failed. This error is reported when no **wfdisc** data are available. If there are no **wfdisc** records for the processing interval, processing can not occur. |
| Action: | See action for "`Error: No wfdiscs found`." |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `Error: Cannot read waveform data` |
| Description: | An error occurred while reading the waveform data from the disk files. |
| Action: | Verify that the disk files exist for the requested waveforms and that read permissions are set to allow the process to access the files. |

| | |
|---|---|
| Message: | `Error: Problem determining processing interval` |
| Description: | The cause of this error is similar to the cause of the error message `Error: Problem determining data interval`. |
| Action: | Follow the actions for `Error: Problem determining data interval`. |

| | |
|---|---|
| Message: | `Error: Cannot QC waveform data` |
| Description: | An error occurred during data quality-control processing. The problem may be a result of inconsistent input data or in the low-level common library code. |
| Action: | Check that the input data are consistent. If inconsistent data are ruled out, contact the application maintainer. |

| | |
|---|---|
| Message: | `Cannot correct waveform data for instrument response` |
| Description: | An error has occurred in the *DFX* library code. |
| Action: | Look for associated errors printed by the library code to assist in identifying the problem. |

| | |
|---|---|
| Message: | `Error: Cannot initialize station` <*STA*> |
| Description: | An error occurred during station initialization. |
| Action: | Review the adjacent error message for the problem cause. |

### Hydro Recall Error Messages

| | |
|---|---|
| Message: | `Fatal error encountered while processing` <*PROCESSING TIME*> |
| Description: | This top-level error message will be printed (along with a more descriptive message) when a major processing error occurs. It will also be printed if all stations or all arrivals being processed encountered an error condition. |
| Action: | Review the associated error messages that follow this one. |

| | |
|---|---|
| Message: | `Cannot correct waveform data for instrument response` |
| Description: | An error has occurred in the *DFX* library code. |
| Action: | Look for associated errors printed by the library code to assist in identifying the problem. |

| | |
|---|---|
| Message: | `Error measuring features for arrival <ARID>` |
| Description: | An error occurred during feature measurement or during output of results to the database. |
| Action: | Check the adjacent errors for more information. If a database output error is the cause, verify that the database is operational and that the target tables exist and have insert permission. |

| | |
|---|---|
| Message: | `Error: Cannot initialize station <STA>` |
| Description: | An error occurred during station initialization. This message should never appear because an error in the station initialization code causes an exception that bypasses this error handler. |
| Action: | Review the adjacent error message for the problem cause. |

| | |
|---|---|
| Message: | `Error: Cannot process arrivals for station <STA>` |
| Description: | All arrivals for this station encountered errors during processing. |
| Action: | Review the adjacent error message for the problem causes. |

| | |
|---|---|
| Message: | `Error: No features to process` |
| Description: | No records were returned from the **hydro_features** database table. |
| Action: | Verify that there are **hydro_features** in the database corresponding to the arrivals being processed, that the **hydro_features** table is properly specified in the par files, and that the database is operational. The **hydro_features** table is set by the parameter *out-hydro_features-table*. |

| | |
|---|---|
| Message: | `Error: No arrivals to process` |
| Description: | No arrival records were found in the database. |
| Action: | Verify that there are arrivals in the input **arrival** table. The **arrival** table is set by the parameter *out-arrival-table*. |

| | |
|---|---|
| Message: | `Inconsistent number of arrival and hydro_features records` |
| Description: | The number of **arrival** and **hydro_features** records do not match. |
| Action: | Verify that the database tables are consistent; if not, resolve the inconsistencies. The **arrival** table is set by the parameter *out-arrival-table*. The **hydro_features** table is set by the parameter *out-hydro_features-table*. |

### Hydro Event Characterization Error Messages

| | |
|---|---|
| Message: | `Fatal error encountered while processing at time` *<PROCESSING TIME>* |
| Description: | This top-level error message is printed (along with a more descriptive message) when a major processing error occurs. |
| Action: | Review the associated error messages for the cause. |

| | |
|---|---|
| Message: | `Error: No sites found` |
| Description: | No **site** records were found in the database for the given network and processing interval. |
| Action: | Verify that the site information is configured properly in the database. The **site** table must contain *sta* values corresponding to the *sta* values for **affiliation**.*net* and the **site**.*ondate* and *offdate* values must bracket the time interval being processed. |

| | |
|---|---|
| Message: | `Error: No sites initialized` |
| Description: | The function organizing station recipes failed for all sites. |
| Action: | Verify that the recipe configurations are correct, and that the top-level directory parameters are set correctly. The top-level configuration file pointed to by the *$IMSPAR* environment variable and the *DFXSPECS* parameter should have the parameter *PARDIR* set to the correct *DFX* configuration root directory. |

| | |
|---|---|
| Message: | `No net time interval computed` |
| Description: | The computation of the processing time interval failed. This problem may be caused by incorrect specification of the TI recipe (see "TI Recipe" on page 77), or by errors in the lower-level *DFX* library code. |
| Action: | Review adjacent error messages for additional information. |

| | |
|---|---|
| Message: | `Error: No wfdatas found` |
| Description: | No sensor channel information was available from the database. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `No origins found for processing` |
| Description: | No origins were returned from the database query. |
| Action: | There may actually be no origins in the processing interval. If there are origins, check that the database is operational and that the correct table, set by the parameter *origin-table*, is being accessed. |

### Interactive Recall Processing Error Messages

| | |
|---|---|
| Message: | `Fatal error encountered while processing` *<PROCESSING TIME>* |
| Description: | This top-level error message is printed (along with a more descriptive message) when a major processing error occurs. It is also printed if all stations or all arrivals being processed encounter an error condition. |
| Action: | Review the error messages associated with this one for the cause. |

| | |
|---|---|
| Message: | `Error getting site results to init travel time tables` |
| Description: | No **site** records were available from the database for the arrival set in the input arrival table. |
| Action: | Verify that the database is operational and contains **site** records for the arrivals contained in the input **arrival** table. The **arrival** table is set by the parameter *out-arrival-table*. |

| | |
|---|---|
| Message: | `Error: No arrivals to process` |
| Description: | No arrival records were found in the database. |
| Action: | Verify that there are arrivals in the input **arrival** table. The **arrival** table is set by the parameter *out-arrival-table*. |

| | |
|---|---|
| Message: | `Error: Cannot initialize <`*TYPE*`> recipes for station <`*STA*`>` |
| Description: | An error occurred reading a processing recipe. |
| Action: | Verify that all recipe configuration files are available and properly configured. Review associated error messages to help identify the problem. |

| | |
|---|---|
| Message: | `Error getting detbeamrecs` |
| Description: | No beam recipes were found having names that match any member of the values in parameter *recall-detection-beams*. |
| Action: | Check the configuration file `DFX-int-recall.par` for the value of *recall-detection-beams*. Check the beam recipe file for the station being processed (see "Interactive Recall Processing" on page 106 for a description of the application's configuration files). |

| | |
|---|---|
| Message: | `Error: Cannot initialize wfdatas for station <STA>` |
| Description: | No sensor channel information was available for the station <*STA*>. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error: Problem determining data interval` |
| Description: | A problem was encountered determining the data interval to process. The data interval is computed using the requested start-times and end-times, and other parameters such as the *data-offset-len*, and the long-term-average length. |
| Action: | Check that the interval parameters are consistent (for example, that the *start-time* is earlier than the *end-time*). |

| | |
|---|---|
| Message: | `Error: No copy of wfdata container can be made` |
| Description: | A copy process failed. This error should only be reported when no **wfdisc** data are available. If there are no **wfdisc**s for the processing interval, processing can not occur. |
| Action: | See action for "`Error: No wfdiscs found`." |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` |
| Description: | No **wfdisc** records were found in the database. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

| | |
|---|---|
| Message: | `Error: Cannot read waveform data` |
| Description: | An error occurred while reading the waveform data from the disk files. |
| Action: | Verify that the disk files exist for the requested waveforms and that read permissions are set to allow the process to access the files. |

| | |
|---|---|
| Message: | `Error: Problem determining processing interval` |
| Description: | The cause of this error is similar to the cause of the error message `Error: Problem determining data interval`. |
| Action: | Follow the actions for `Error: Problem determining data interval`. |

| | |
|---|---|
| Message: | `Error: Cannot QC waveform data` |
| Description: | An error occurred during data quality-control processing. The problem may be a result of inconsistent input data or in the low-level common library code. |
| Action: | Check that the input data are consistent. If inconsistent data are ruled out, contact the application maintainer. |

| | |
|---|---|
| Message: | `Error getting site results to init travel time tables` |
| Description: | No **site** records were available from the database for the arrival set in the input **arrival** table. |
| Action: | Verify that the database is operational and contains **site** records for the arrivals contained in the input **arrival** table. The **arrival** table is set by the parameter *out-arrival-table*. |

| | |
|---|---|
| Message: | `Error reading det object for arrival` <*ARID*> |
| Description: | An error occurred in the `DFXdefault.scm` function *create-recall-det-from-dbarrival*. This error should never occur. |
| Action: | Contact the application maintainer. |

| | |
|---|---|
| Message: | `Error finding best beam for arrival` <*ARID*> |
| Description: | The application was unable to determine the detection beam with the best snr. This problem could be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection, causing errors during the snr computation. It could also be a result of a configuration error in which no detection beams were specified in the beam recipe, although this condition should be caught as: `Error getting detbeamrecs`. |
| Action: | Verify that the detection beams are specified in the beam recipe. |

| | |
|---|---|
| Message: | `Error revising estimates for arrival` <*ARID*> |
| Description: | An error occurred during feature measurement computations. |
| Action: | See the error messages that follow this message. |

| | |
|---|---|
| Message: | `Error measuring deltim for detection at time` <*TIME*> |
| Description: | This error will most likely occur when the detection recipe parameters *det-max-snr*, *det-min-snr*, *det-max-deltime*, or *det-min-deltime* have unreasonable values (for example, *det-min-deltime* > *det-max-deltime*). |
| Action: | Check and correct unreasonable recipe parameter values. |

---

Message: `Error measuring amplitude for best beam for detection at time <TIME>`

Description: The amplitude measurement computation failed. This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection.

Action: Review the preceding log messages for more information.

---

Message: `Error measuring fk for detection at time <TIME>`

Description: The FK processing failed. An error occurred in a *DFX* or common library function.

Action: Review the preceding log messages for more information.

---

Message: `Error measuring amplitude for fk beam for detection at time <TIME>`

Description: The amplitude measurement computation failed. This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection.

Action: Review the preceding log messages for more information.

---

Message: `Error measuring polarization for detection at time <TIME>`

Description: The particle motion analysis computation failed. (1) This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection. (2) Particle motion analysis requires 3-C, and all components may not be available. (3) An error may have occurred in the *DFX* library code.

Action: Review the preceding log messages for more information.

---

| | |
|---|---|
| Message: | `Error measuring amplitude <AMPREC-NAME> for detection at time <`*TIME*`>` |
| Description: | The amplitude measurement computation failed. This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection. |
| Action: | Review the preceding log messages for more information. |

### QC Stats Error Messages

| | |
|---|---|
| Message: | `list-to-list-pair: input list does not have an even number of elements.` |
| Description: | The parameter *qcstats-cull-sta-chan-list* is poorly formed. This parameter should contain a list of station/channel pairs in the form of `STA1,CHAN1,...` (for example: `qcstats-cull-sta-chan-list= AR01,sz,AR01,sn`). |
| Action: | Correct errors in the parameter *qcstats-cull-sta-chan-list*. |

| | |
|---|---|
| Message: | `ERROR IN QCSTATS` |
| Description: | This top-level error message is printed (along with a more descriptive message) when a major processing error occurs. |
| Action: | Review the associated error messages for the cause. |

| | |
|---|---|
| Message: | `DFX-qcstats.scm(write-qcstats) Error: could not submit qcstats to database!` |
| Description: | An error occurred during the **qcstats** database insert. |
| Action: | Verify that the database is operational, and that the **qcstats** table exists (parameter *out-qcstats-table*) and has insert permission. |

| | |
|---|---|
| Message: | `DFX-qcstats.scm(process-qcstats) Error: no data in data-con` |
| Description: | The data container provided by the parent application contained no data. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

### Seismic Recall Error Messages

| | |
|---|---|
| Message: | `Fatal error encountered while processing at time` *<PROCESSING TIME>* |
| Description: | This top-level error message is printed (along with a more descriptive message) when a major processing error occurs. |
| Action: | Review the associated error messages for the cause. |

| | |
|---|---|
| Message: | `Error: No arrivals to process` |
| Description: | No **arrival** records were found in the database. |
| Action: | Verify that there are qualifying arrivals in the input **arrival** table. The **arrival** table is set by the parameter *out-arrival-table*. To satisfy this application's query, the arrivals must satisfy the following conditions:<br>1. The arrivals are associated with origins occurring within the specified processing interval.<br>2. The arrivals have snr values less than 0.<br>3. The arrivals do not have **assoc**.*phase* values of "LR."<br>4. The arrivals are part of the **affiliation** network specified by the *net* parameter. |

| | |
|---|---|
| Message: | `Error initializing for residual update` |
| Description: | No records were returned by queries to the **assoc**, **site**, or **origin** tables. Data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you feel that data are available, verify that the correct database is being accessed and that the correct tables are being used. The table names are specified by the parameters *in-assoc-table*, *site-table*, and *origin-table*. |

| | |
|---|---|
| Message: | `Error computing beam for beam` *<BEAM NAME>* |
| Description: | An error occurred either during beam computation in *DFX* or in a common library function. |
| Action: | Review the associated error messages to obtain detailed information about the problem. |

| | |
|---|---|
| Message: | `Error computing snr for beam` <*BEAM NAME*> |
| Description: | Either an error occurred during snr computation in *DFX* or in a common library function, or no snr measure was computable for this beam at the detection time. Typical causes of this error are poor quality waveform data (resulting in excessive data quality-control masking) and significant numbers of unavailable data channels. This, in turn, can cause the number of channels to fall below the threshold computed using the *det-min-sta-fraction* parameter. See "Table 12: Detection Processing Parameters" on page 50. |
| Action: | Examine the quality of the data to determine whether or not excessive masking caused the error. If so, then no action is necessary; if not, then contact the application developer. |

| | |
|---|---|
| Message: | `Error measuring deltim for detection at time` <*TIME*> |
| Description: | This error will most likely occur when the detection recipe parameters *det-max-snr*, *det-min-snr*, *det-max-deltime*, or *det-min-deltime* have unreasonable values (for example, *det-min-deltime* > *det-max-deltime*). |
| Action: | Correct unreasonable values in the detection recipe parameters. |

| | |
|---|---|
| Message: | `Error measuring amplitude for best beam for detection at time` <*TIME*> |
| Description: | The amplitude measurement computation failed. This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection. |
| Action: | Review the preceding log messages for more information. |

| | |
|---|---|
| Message: | `Error measuring fk for detection at time` <*TIME*> |
| Description: | The *FK* processing failed. An error occurred in a *DFX* or common library function. |
| Action: | Review the preceding log messages for more information. |

| | |
|---|---|
| Message: | `Error measuring amplitude for fk beam for detection at time` <*TIME*> |
| Description: | The amplitude measurement computation failed. This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection. |
| Action: | Review the preceding log messages for more information. |

| | |
|---|---|
| Message: | `Error measuring polarization for detection at time` <*TIME*> |
| Description: | The particle motion analysis computation failed. (1) This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection. (2) Particle motion analysis requires 3-C data, and all components may not be available. (3) An error may have occurred in the *DFX* library code. |
| Action: | Review the preceding log messages for more information. |

| | |
|---|---|
| Message: | `Error making dbarrival for detection at time` <*TIME*> |
| Description: | An error occurred while constructing an arrival object from the contents of a detection object. This error is caused by problems in the *DFX* library code. |
| Action: | See associated error messages to obtain more information about the cause. |

| | |
|---|---|
| Message: | `Error measuring amplitude` <*AMPREC-NAME*> `for detection at time` <*TIME*> |
| Description: | The amplitude computation failed during the *depth-phase-snr* measurement. This failure may be due to extensive data quality-control masking of the waveform data in the neighborhood of the detection. |
| Action: | Review the preceding log messages for more information. |

| | |
|---|---|
| Message: | `Error making depth-phase dbamplitude for amplitude` <*AMPREC-NAME*> `for detection at time` <*TIME*> |
| Description: | An error occurred converting the amplitude measurement to a dbamplitude object. This error is caused by problems in the *DFX* library code. |
| Action: | Look for associated error messages to obtain more information about the cause |

### Segment Archiving Error Messages

| | |
|---|---|
| Message: | `Error: No dbsites found` <*QUERY*> |
| Description: | No array (**site.***statype* **=** `ar`) **site** records were found in the database for the given network and processing interval. |
| Action: | Verify that the site information is configured properly in the database. The **site** table must contain *sta* values corresponding to the *sta* values for **affiliation**.*net* and the **site**.*ondate* and *offdate* values must bracket the time interval being processed. |

| | |
|---|---|
| Message: | `Error: No dbarsocs found` <*QUERY*> |
| Description: | No associated arrivals were found that satisfy the printed <*QUERY*>. |
| Action: | Check for proper configuration of the database. |

| | |
|---|---|
| Message: | `Error: No dborigins found` <*QUERY*> |
| Description: | No origins were found that satisfy the printed <*QUERY*>. There may be no origins for processing. |
| Action: | Check that origins exist in the processing interval; if so, verify that the correct table is being accessed. |

| | |
|---|---|
| Message: | `Error: No wfdatas found` |
| Description: | No sensor channel information was available from the database. |
| Action: | Verify that the station being processed is configured properly in the database. Check all links between the tables **site, sitechan, sensor, instrument**, and **affiliation**. |

| | |
|---|---|
| Message: | `Error forming channel string for wfdisc query` |
| Description: | No channels were found in the wfdata (**site**, **sitechan**) query results. |
| Action: | Check that the database is operational and that all the proper tables are being accessed. |

| | |
|---|---|
| Message: | `Error: No wfdiscs found` <*QUERY*> |
| Description: | No **wfdisc** records were found in the database for the <*QUERY*>. Waveform data may not be available for the time period requested, in which case no processing can take place. |
| Action: | If you believe that waveform data are available, verify that the correct database is being accessed and that the correct input **wfdisc** table is being used. The default input **wfdisc** table name is `wfdisc`, but can be set explicitly with the parameter *in-wfdisc-table*. |

### Informational Messages

Examining a *DFX* log file for errors can be daunting because of the voluminous output. Some log entries may appear to indicate errors, but are usually part of the standard logfile output during successful processing. The following log entries are common informational messages. No actions are necessary for these messages.

| | |
|---|---|
| Message: | `*** unable to find file ".DFXinit" in path` <*PATH*> `***` |
| Description: | This message is informational. *DFX* loads the file `.DFXinit` in the user's home directory. If no `.DFXinit` is present, then this message is printed. |

| | |
|---|---|
| Message: | `get_amplitude_recipe: No beam recipe specified for amplitude recipe ARRIVAL` |
| Description: | This message is printed when the amplitude recipe specifies the measurement beam name as "`-`" (the NULL value). However, the ARRIVAL amplitude recipe is for measurements on the arrival beam, which is determined during processing, so this message may be ignored. |

| | |
|---|---|
| Message: | `gobj_qc_waveforms: Number of components 1 <= minimum 2. Using extended single station masking` |
| Description: | This data quality-control processing message states that single-component data quality-control processing is being used because the number of available components is below the threshold (see the *qc-min-multi-component* parameter of the "QC Recipe" on page 67). |

| | |
|---|---|
| Message: | `gobj_qc_waveforms: Number of components 19 > minimum 2. Using extended array masking` |
| Description: | This data quality-control processing message states that multi-component processing is being used. |

| | |
|---|---|
| Message: | `Deleting detection at time <TIME1> due to detection at time <TIME2>` |
| Description: | Two detections were made at an interval less than the *det-min-detection-interval* value (see "Detection Recipe" on page 50). |

| | |
|---|---|
| Message: | `Info: No objects found for query SELECT * FROM arrival WHERE time BETWEEN <TIME1> AND <TIME2> AND sta=<STA>` |
| Description: | This query identifies detections during the interval so that duplicate detections are not written. The message indicates that this interval has probably not been processed. |

| | |
|---|---|
| Message: | `screen_half_periods: Info: period <PERIOD> exceeds the <DURATION> second window length percentage threshold.` |
| Description: | This message is being printed because the period computed during the amplitude processing is larger than allowable. This is probably a data-related, rather than processing-related, issue. |

| | |
|---|---|
| Message: | `calc_best_period: Warning: Ratio of max and min half periods 3.180 exceeds threshold of 2.000` |
| Description: | The period values are too different for the adjacent multiple half periods. This is probably a data-related, rather than processing-related, issue. |

| | |
|---|---|
| Message: | `"SBSNR < 1 for chan <CHAN> arid <ARID>" "Warning. No sbsnr dbamplitude made for detection at time <TIME>"` |
| Description: | The particular SBSNR beam had an snr value of less than 1.0. This is probably a data-related, rather than processing-related, issue. |

| | |
|---|---|
| Message: | `Nothing to put into table <TABLE>` |
| Description: | An informational message printed when the processing did not create any output for the particular table. |

## SOLVING COMMON PROBLEMS

Occasionally an interactive *DFX* process returns an error status to *ARS*. Unfortunately, the error status is reported simply as a failure without any information as to what the failure is or what may have caused it.

The causes for processing failures are most likely related to problems with the input data, rather than program errors. For example, poor quality signals may be excessively masked by the data quality-control process, which render the required features in the **interval** uncomputable.

To investigate the cause of a processing failure, review the application log file. The log file locations are provided in "Chapter 2: Operational Procedures" on page 19. For example, the *Interactive Recall* processing log file location is described in "Log File Location" on page 106.

### Error Recovery

*DFX* applications are well behaved. If processing is interrupted, the same time interval can be re-processed without deleting the results of the previous processing.

## REPORTING PROBLEMS

The following procedures are recommended for reporting problems with the application software:

1. Diagnose the problem as far as possible.
2. Record information regarding symptoms and conditions at the time of the software failure.
3. Retain copies of relevant sections of application log files.
4. Contact the provider or maintainer of the software for problem resolution if local changes of the environment or configuration are not sufficient.

# Chapter 4: Installation Procedures

This chapter provides instructions for installing the software and includes the following topics:

# Chapter 4: Installation Procedures

## PREPARATION

### Obtaining Released Software

The software is obtained via FTP from a remote site or via a physical medium, such as tape or CD-ROM. The software and associated configuration data files are stored as one or more tar files. The software and data files are first transferred via FTP or copied from the physical medium to an appropriate location on a local hard disk. The tar files are then untarred into a standard UNIX directory structure.

### Hardware Mapping

The user must select the hardware on which to run the software components. Software components are generally mapped to hardware to be roughly consistent with the software configuration model.

### UNIX System

No special operating system configuration is necessary for running the *DFX* applications.

## EXECUTABLE FILES

The *DFX* executable should be installed in the binary directory along with other system executable files.

## CONFIGURATION DATA FILES

The configuration directory structure is flexible. The application configuration files described in "Maintenance" on page 38 obtain key directory locations from the top-level system parameter file specified by the parameter or environment variable *IMSPAR*. Typical locations for the parameter files are also described in "Maintenance" on page 38.

## DATABASE

The *DFX* applications database interactions are described in detail in [IDC7.1.1].

### Accounts

The database accounts required for the *DFX* applications are specified as part of their runtime parameters.

### Tables

*DFX* detection processing utilizes a station lock table to preclude a race condition while screening for existing detections. The default name for this table is `DFX$DETECTION$LOCK`, but the name may be specified by the parameter *dfx-lock-table*. The table structure is a single column containing a list of stations that *DFX* processes. (See [IDC5.1.1Rev3].)

### Initialization of LastID

*DFX* applications use the following keys from the **lastid** table: *ampid*, *arid*, *parid*, *qcstatsid*, and *wfid*. Appropriate initial values should be set such that there are no key collisions with existing data sets.

## TUXEDO FILES

The *tuxshell* configuration files associated with the *DFX* applications are described in "Maintenance" on page 38.

## INITIATING OPERATIONS

Methods for running *DFX* applications are described in "Software Startup" on page 20.

## VALIDATING INSTALLATION

Test data intervals for which ground truth processing results exist should be run through the *DFX* applications. The *DFX* output should be consistent with the previous values.

# References

The following sources supplement or are referenced in the document:

[IDC5.1.1Rev3] Science Applications International Corporation, Veridian Systems, *Database Schema, (Part 1, Part 2, and Part 3), Revision 3*, SAIC-01/3052, TN-2866, 2001.

[IDC5.2.1Rev1] Science Applications International Corporation, *IDC Processing of Seismic, Hydroacoustic, and Infrasonic Data, Revision 1*, 2001.

[IDC6.2.4Rev1] Science Applications International Corporation, *Configuration of IDC Processing Data Files, Revision 1*, 2001.

[IDC6.5.2Rev0.1] Science Applications International Corporation, *Distributed Application Control System (DACS) Software User Manual, Revision 0.1*, SAIC-00/3038, 2000.

[IDC7.1.1] Science Applications International Corporation, *Detection and Feature Extraction (DFX)–Scheme Files*, SAIC-01/3000, 2001.

[IDC7.4.1] Science Applications International Corporation, *Continuous Data Subsystem CD–1.1*, SAIC-01/3012, 2001.

[Jep98] Jepsen, D., *PIDC 6.0: Event Characterization*, CCB-PRO-98/07, 1998.

[Kvæ95] Kværna, T., "Automatic Onset Time Estimation Based on Autoregressive Processing," *Semiannual Technical Summary, 1 April–30 September 1995*, NORSAR Scientific Report 1–95/96, Kjeller, Norway, pp. 113–133, 1995.

▼ **References**

[Lan97g] Laney, H., *Hydroacoustic Feature Extractor (DFX-H Version 1) Technical Description*, CMR-97/06, 1997.

[Sle78] Slepian, D., "Prolate Spheroidal Wave Functions, Fourier Analysis and Uncertainty - V: The Discrete Case," *Bell Systems Technical Journal*, Volume 57, No. 5, pp. 1371–1430, 1978.

[Tho82] Thomson, D. J., "Spectrum Estimation and Harmonic Analysis," *Proceedings of IEEE*, Volume 70, No. 9, pp. 1055–1096, 1982.

# Glossary

Selected terms in this glossary are specific to this document and are not applicable in any other context. These glossary items are indicated by appending "(*DFX* only)" to the term.

## Symbols

**1-C**

One-component; a single component of motion.

**3-C**

Three-component.

## A

**AIC**

Akaike Information Criterion.

**Akaike Information Criterion**

A mathematical method for distinguishing between noise and signal. Used in estimating onset times. See [IDC5.2.1Rev1]

**amp**

Amplitude.

**amplitude**

Zero-to-peak height of a waveform in nanometers.

**Analyst Review Station**

This application provides tools for a human analyst to refine and improve the event bulletin by interactive analysis.

**application (DACS, Tuxedo)**

System of cooperating processes configured for a specific function to be run in a distributed fashion by Tuxedo. Also used in a more general sense to refer to all objects included in one particular `ubbconfig` file (machines, groups, servers) and associated shared memory resources, qspaces, and clients.

**array**

Collection of sensors distributed over a finite area (usually in a cross, triangle, or concentric pattern) and referred to as a single station.

**arrival-based amplitude**

Amplitude measured by *DFX* for a detected signal.

**ARS**

See Analyst Review Station.

**attribute**

(1) Database column. (2) Characteristic of an item; specifically, a quantitative measure of a S/H/I detection such as azimuth, slowness, period, or amplitude.

**azimuth**

Direction, in degrees clockwise with respect to North, from a station to an event.

## B

**B**

Byte.

**beam**

(1) Waveform created from array station elements that are sequentially summed after being steered to the direction of a specified azimuth and slowness. (2) Any derived waveform (for example, a filtered waveform).

**beam recipe**

Table of parameters that describe how to create and use a beam. Parameters may include the channels to use in forming the beam, the filter to use, the direction and slowness in which to steer the beam, and the snr threshold to apply for detection.

**beam-on-the-fly**

*DFX* Scheme application that creates origin beams and can be called from within *ARS*.

**Beamer**

Application that prepares origin beams for interactive analysis.

**beamform**

Sum a set of waveforms from array station elements with time delays introduced to compensate for the time it takes a wave to travel across the array.

## C

**cepstrum (cepstral)**

Fourier transformation of a power spectrum whose magnitudes have been scaled logarithmically.

**channel**

Component of motion or distinct stream of data.

**command**

Expression that can be input to a computer system to initiate an action or affect the execution of a computer program.

**component**

(1) One dimension of a three-dimensional signal; (2) The vertically or horizontally oriented (north or east) sensor of a station used to measure the dimension; (3) One of the parts of a system; also referred to as a module or unit.

**Computer Software Component**

Functionally or logically distinct part of a computer software configuration item; possibly an aggregate of two or more software units.

**Computer Software Configuration Item**

Aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

**CSC**

See Computer Software Component.

**CSCI**

See Computer Software Configuration Item.

## D

**DACS**

See Distributed Application Control System.

**dB**

Decibel.

**db<tablename>** (*DFX* only)

Generic object with fields that have a one-to-one correspondence with the database table attributes.

**deg.**

Degrees (as a distance).

**detection**

Probable signal that has been automatically detected by the Detection and Feature Extraction (*DFX*) software.

**Detection and Feature Extraction**

*DFX* is a programming environment that executes applications written in Scheme (known as *DFX* applications).

**detection beam**

Same as an f-k beam.

**Detection Processing**

Automatic Seismic Detection processing. A *DFX* Scheme application that makes detections and measures features on seismic waveform data.

**DFX**

See Detection and Feature Extraction.

**Distributed Application Control System**

This software supports inter-application message passing and process management.

## E

**element**

(1) Single station or substation of a sensor array, referred to by its element name (such as YKR8), as opposed to its array name (YKA in this example). (2) Data storage location in a data array.

**epoch time**

Number of seconds after January 1, 1970 00:00:00.0.

**event**

Unique source of seismic, hydroacoustic, or infrasonic wave energy that is limited in both time and space.

**event characterization**

IDC process of characterizing events by features of signals recorded at one or more stations.

**execute**

Carry out an instruction, process, or computer program.

## F

**f-k**

Frequency (f) versus wavenumber (k) analysis that maps phase power from an array as a function of azimuth and slowness.

**f-k beam**

Coherent beam steered to the azimuth and slowness of the maximum peak in an f-k spectrum.

**f-k spectrum**

Result of a data transformation from the time-space domain to the frequency-wavenumber domain. Useful in determining the direction and slowness of an arriving phase.

**F-statistic**

Measure that indicates the degree of spatial coherence of a waveform across an array of sensors. This measure is approximately equal to the ratio of the spatially coherent energy to the incoherent energy scaled by the number of non-collocated sensors.

**features**

Various measurements of a waveform segment used to characterize a detection.

**FIR**

Finite Impulse Response (usually in reference to a filter).

**FTP**

File Transfer Protocol; protocol for transferring files between computers.

**function**

Named section of a program that performs a particular task.

## G

**generic object**

Construct used to hold and manipulate data. The type of object determines the data that it can contain. Also known as an object or GObj.

**GMT**

Greenwich Mean Time.

**GObj**

See generic object.

## H

**hydroacoustic**

Pertaining to sound in the ocean.

**Hydro Detection**

Automatic Hydroacoustic Detection processing. A *DFX* Scheme application that makes detections and measures features on hydroacoustic waveform data.

**Hydro Event Characterization**

Automatic Hydroacoustic Event Characterization processing. A *DFX* Scheme application that computes hydroacoustic measures relative to events typically formed by the seismic network. It can be used to rule out the possibility that the event occurred in water.

**Hz**

Hertz.

## I

**Infra Detection**

Automatic Infrasonic Detection processing. A *DFX* Scheme application that makes detections and measures features on infrasonic waveform data.

**infrasonic (infrasound)**

Pertaining to low-frequency (sub-audible) sound in the atmosphere.

**Hydro Recall**

Interactive Hydroacoustic Recall processing. A *DFX* Scheme application that is called from *ARS* to measure features on analyst modified or added hydroacoustic detections.

**Interactive Recall Processing**

Interactive Seismic Recall processing. A *DFX* Scheme application that is called from *ARS* to measure features on analyst modified or added seismic detections.

**IPC**

Interprocess communication. The messaging system by which applications communicate with each other through *libipc* common library functions. See *tuxshell*.

## J

**jdate**

Modified Julian Date. Concatenation of the year and three-digit Julian day of year. For example, the jdate for 07 March, 2000, is 2000067.

**Julian date**

Increasing count of the number of days since an arbitrary starting date.

## K

**KB**

Kilobyte. 1,024 bytes.

**km**

Kilometer.

## L

**LP**

Long period.

## M

**mask**

Array of start and end indices for defective data samples in a time series used to later identify the defective data.

**MB**

See megabyte.

**megabyte**

1,024 kilobytes.

**min**

Minute.

## N

**N/A**

Not Applicable.

**network**

Spatially distributed collection of seismic, hydroacoustic, or infrasonic stations for which the station spacing is much larger than a wavelength.

**noise**

Incoherent natural or artificial perturbations of the waveform trace caused by ice, animals migrations, cultural activity, equipment malfunctions or interruption of satellite communication, or ambient background movements.

**Noise Amplitude**

Automatic Noise Amplitude Estimation. A *DFX* Scheme application that measures the noise level at stations that did not detect signals from a given event.

**NSE**

Noise Spectrum Equalization.

**NULL**

Empty, zero.

**Nyquist frequency**

Minimum number of counts needed to define a particular frequency.

## O

**ORACLE**

Vendor of the database management system used at the PIDC and IDC.

**orid**

Origin Identifier.

**origin**

Hypothesized time and location of a seismic, hydroacoustic, or infrasonic event. An event may have many origins. Characteristics such as magnitudes and error estimates may be associated with an origin.

**origin beam**

Coherent beam steered to the estimated event origin.

**Origin Beam**

Automatic Origin Beam processing. A *DFX* Scheme application that computes and saves origin beams for use in interactive analysis.

## P

**par**

See parameter.

**parameter**

User-specified token that controls some aspect of an application (for example, database name, threshold value). Most parameters are specified using [*token* = *value*] strings, for example, `dbname=mydata/base@oracle`.

**parameter (par) file**

ASCII file containing values for parameters of a program. Par files are used to replace command line arguments. The files are formatted as a list of [*token* = *value*] strings.

**phase**

Arrival that is identified based on its path through the earth.

**phase name**

Name assigned to a seismic, hydroacoustic or infrasonic arrival associated with a travel path.

**polarization**

Form of three-component analysis used to derive azimuth and slowness information from non-array stations.

**polarization analysis**

Analysis to determine the propagation vector that describes particle motion. Used at seismic 3-C stations.

**process**

Function or set of functions in an application that perform a task.

**processing unit**

Software component of a larger entity such as a program.

**program**

Organized list of instructions that, when executed, causes the computer to behave in a predetermined manner. A program contains a list of variables and a list of statements that tell the computer what to do with the variables.

## Q

**QC**

Quality Control.

**QC Stats**

Quality Control Statistics. A *DFX* Scheme application that computes and saves a variety of statistics related to the input waveform data quality.

**quefrency**

Time-delay axis with units of seconds for a cepstrum.

**query**

Request for specific data from a database.

## R

**recall processing**

Type of *DFX* processing that updates arrival features based on changes made during analyst review.

**recipe**

Collection of related parameters used in performing a specific task. Recipe parameters are usually defined in par files.

**run**

(1) Single, usually continuous, execution of a computer program. (2) To execute a computer program.

## S

**s**

Second(s) (time).

**S/H/I**

Seismic, hydroacoustic, and infrasonic.

**SAIC**

Science Applications International Corporation.

**schema**

Database structure description.

**Scheme**

Dialect of the Lisp programming language that is used to configure some IDC software.

**Segment Archiving**

A *DFX* Scheme application that extracts, beamforms, and stores waveform data relevant to an event (occurs after analyst review).

**seismic**

Pertaining to elastic waves traveling through the earth.

**Seismic Event Characterization**

Automatic Seismic Event Characterization processing. A *DFX* Scheme application that makes measurements on seismic waveforms in a time window predicted to contain signals from an event in order to characterize the event.

**Seismic Recall**

Automatic Seismic Recall processing. A *DFX* Scheme application run in post-analysis processing.

**SIOD**

Scheme In One Defun (version of Scheme).

**snr**

Signal-to-noise ratio.

**Solaris**

Name of the operating system used on Sun Microsystems hardware.

**SP**

Short period.

**splp**

Short-period to long-period amplitude ratio used in event characterization.

**SQL**

Structured Query Language; a language for manipulating data in a relational database.

**station**

Collection of one or more monitoring instruments. Stations can have either one sensor location (for example, BGCA) or a spatially distributed array of sensors (for example, ASAR).

**station processing**

Processing based on data from a single station.

**structure**

Software construct that collects one or more variables, possibly of different types, together under a single name for convenient handling.

**subprocess**

Function or set of functions that perform a task subordinate to a process.

**subsystem**

Secondary or subordinate system within the larger system.

## T

**TF** (*DFX* only)

Time Frequency analysis.

**third moment of frequency**

Measure of the frequency content of a signal. Used in event characterization.

**TI**

Class of DACS servers that form time intervals by station sensor (for example, *tis_server*).

**TI recipe**

Time Interval recipe. Parameters used by *DFX* to specify a time interval of data to read or process.

**TM**

Threshold monitoring. A technique to keep track of the minimum detectable event based on noise levels at stations.

**TMF**

See third moment of frequency.

**trigger**

(1) Action a database should take when some database-related event occurs. (2) Object used in *DFX* to indicate a potential transient signal for a given beam. Triggers for a set of beams are collected to form detections.

**Tuxedo**

Transactions for UNIX Extended for Distributed Operations.

**tuxpad**

DACS client that provides a graphical user interface for common Tuxedo administrative services.

**tuxshell**

Process in the Distributed Processing CSCI used to execute and manage applications. See IPC.

## U

**UNIX**

Trade name of the operating system used by the Sun workstations.

## V

**Velocity Model Specification File**

File for setting travel-time models in S/H/I event location.

**VMSF**

See Velocity Model Specification File.

## W

**waveform**

Time-domain signal data from a sensor (the voltage output) where the voltage has been converted to a digital count (which is monotonic with the amplitude of the stimulus to which the sensor responds).

**wavenumber**

Vector, *k*, in the direction of a propagating wave whose magnitude is given by the inverse wavelength of the wave scaled by a factor of $2\pi$.

**wfdisc**

Waveform description record or table.

# Index

## A



## B

## C



## D

## E



## F



## G

## H



## I

## L



## M



## N



## O

## P



## Q



## R

## S

## T

## U



## X